Available online at www.sciencedirect.com
SCIENCE DIRECT®
Comput. Methods Appl. Mech. Engrg. 192 (2003) 3343–3377
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A Chimera method based on a Dirichlet/Neumann(Robin) coupling for the Navier–Stokes equations

Guillaume Houzeaux, Ramon Codina *

Universitat Politècnica de Catalunya, Jordi Girona 1-3, 08034 Barcelona, Spain

Received 25 February 2002; received in revised form 3 March 2003; accepted 10 March 2003

Abstract

We present a Chimera method for the numerical solution of incompressible flows past objects in relative motion. The Chimera method is implemented as an iteration-by-subdomain method based on Dirichlet/Neumann(Robin) coupling. The DD method we propose is not only geometric but also algorithmic, for the solution on each subdomain is obtained on separate processes and the exchange of information between the subdomains is carried out by a master code. This strategy is very flexible as it requires almost no modification to the original numerical code. Therefore, only the master code has to be adapted to the numerical codes and the strategies used on each subdomain. As a basic flow solver, we a use stabilized finite element method.

Keywords: Mixed domain decomposition method; Stabilized finite elements; Chimera; Iteration-by-subdomain; Incompressible flows

1. Introduction

When one wants to simulate flows with moving bodies and when there is no possible way of prescribing simple boundary conditions in any frame of reference, four main alternatives to track the body motions are possible:

- the arbitrary-Lagrangian–Eulerian (ALE) method together with an automatic remeshing technique of the computational domain adapts the fluid mesh to the spatial configuration in time;
- the fictitious domain method tracks moving solid boundaries inside a background mesh;
- the sliding mesh technique couples different meshes which are allowed to slide along their common interfaces;
- the Chimera method couples the individual meshes of each moving component.

* Corresponding author.
E-mail address: codina@cimne.upc.es (R. Codina).

0045-7825/03/$ - see front matter
doi:10.1016/S0045-7825(03)00276-7

These techniques are illustrated in Fig. 1.

When using the ALE description of the flow together with automatic remeshing, the mesh accommodates the boundary displacements inside the computational domain (see for example [1–3]). On the one hand, if the displacements are small, only nodal displacement may be sufficient, and the nodal connectivity of the mesh remains unchanged; on the other hand, if the displacements are large, a complete remeshing is necessary. The main drawback of the method is that the geometric parameters have to be computed at each time iteration. See [4] for an example of application to the simulation of a mixed-flow pump. The ALE technique has also been used for following free surfaces [5,6] and to simulate fluid/structure interactions [7].

In the fictitious domain method [8,9], a fixed mesh occupies the whole volume including that occupied by the body. The method consists in including the boundary condition at the body boundary into the set of flow equations for the whole volume by the way of Lagrange multipliers. In the particular case of Dirichlet conditions imposed on the body, the Lagrange multiplier represents the jump in traction obtained at the fluid–solid interface. This method enables one to use simple (structured) background meshes on which fast solvers can be implemented. In [10], a fictitious domain method is presented to simulate two- and three-dimensional flow problems with moving boundaries. The authors apply the method to the solution of a Couette problem and a helical ribbon mixer; in [11], the fictitious method is applied to the solution of the flow around a moving disk. In fictitious domain methods, the motion of the object needs not to be known a priori, and aerodynamic forces can be taken into account to couple the fluid dynamics and the kinematics of the rigid body. Pan [12] predicts the path of a ball falling in a viscous fluid (at low Reynolds numbers); in [13], the authors solve the two-dimensional flow around an airfoil that is free to rotate around its center of mass, the sedimentation of particles in a box, and a three-dimensional case involving two spherical particles. Using the same method, Juárez [14] simulates the sedimentation of an elliptic body in a two-dimensional viscous fluid. This fictitious domain method is also well suited for shape optimization problems [15]. Although it has been shown that this method can efficiently solve the flow over moving objects, it presents a serious drawback: at large Reynolds numbers, we have no simple way to refine the mesh near the boundary without dropping the nice characteristic of the method. A possible technique is to use a *h*-type refinement consisting in dividing successively elements of the region of interest into smaller and smaller elements.

The sliding mesh technique regroups DD methods for which two adjacent subdomains are allowed to slide along their common interface. In this work, we generalize this technique to any DD method involving possibly moving overlapping subdomains for which the interface topologies do not change with time. As it

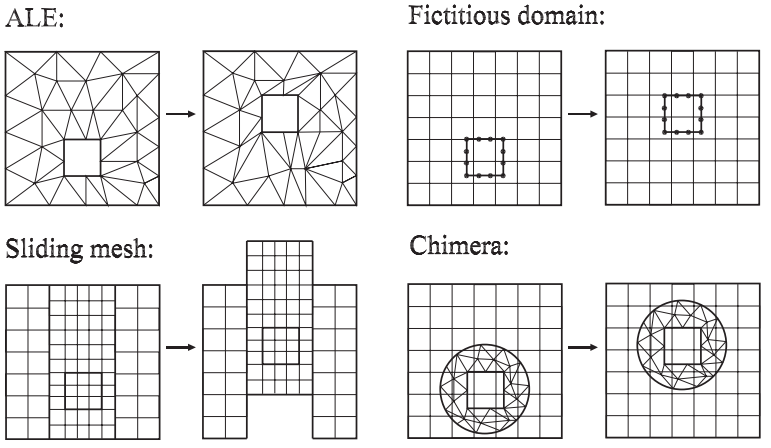

Fig. 1. Illustration of the most common methods to simulate flows around moving components.

can be a hard task to ensure that the nodes of two adjacent sliding meshes coincide at each time step, the sliding mesh method is generally used as a direct application of the mortar method [16,17] to moving subdomains [18]. The mortar element method is a non-conforming domain decomposition method for coupling non-matching grids. Instead of considering the continuity of the transmission variables point by point by using a simple interpolation technique, the mortar element method performs an interface L^2-projection of the transmission conditions. When the mortar method is used together with a sliding mesh technique, the subdomains are allowed to slide along their common interface. They are therefore necessarily disjoint. See [19] for the application of the sliding mesh technique to the simulation of stirred reactors, and [20] for the simulation of a two-dimensional rotor–stator interactions in a centrifugal pump. We also mention the shear-slip mesh update method [21] (SSMUM) where regions in relative straight line translation or rotation are glued by the way of intermediate layers of elements, and where the connecting nodes coincide. In order to avoid remeshing of the regions, only the elements of the intermediate layer are allowed to be deformed and its computational domain to be remeshed when necessary. The advantage of this method is that it is conservative as the composite mesh is always conforming. The main drawback is that arbitrary motions are not possible.

The Chimera method appears to be the most flexible method to treat flow problems with moving bodies [22–24]. In addition, it fits perfectly within the Chimera based iteration-by-subdomain method introduced in this work. It was first envisaged as a tool for simplifying the mesh generation [39–41]. Independent meshes are generated for each component of the computational domain, enabling a flexibility on the choice of the type of element as well as on their orientation that could not be possible when meshing complex three-dimensional geometries [42,43]. As a direct application, the Chimera method has also been used as a mesh refinement technique [44]. In addition, if it is implemented efficiently, it is a very efficient tool to treat flows with moving components [45]. Traditionally, the coupling is performed using a Dirichlet/Dirichlet iteration-by-subdomain method. In this work we extend this to include also the possibility of using Dirichlet/Neumann and Dirichlet/Robin couplings. We present a complete description of the formulation of this method as well as of its implementation aspects, focusing on the treatment of flows with moving bodies.

2. Domain decomposition for stationary the Navier–Stokes equations

2.1. Problem statement: incompressible flow equations

We now give the expression of the Navier–Stokes equations expressed in a non-inertial frame of reference [25]. We denote $\boldsymbol{a}$ as the linear acceleration of the frame of reference and $\boldsymbol{\omega}$ as its angular velocity. Let $\boldsymbol{g}$ be the gravitational acceleration and $\boldsymbol{x}$ the position vector of a fluid point. Let Ω be an open bounded domain of $\mathbb{R}^{n_d}$ ($n_d = 2$ or 3). The stationary Navier–Stokes equations for the velocity $\boldsymbol{u}$ and the pressure p are:

$$(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + 2\boldsymbol{\omega}\times\boldsymbol{u} - 2\nu\nabla\cdot\boldsymbol{\varepsilon}(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \quad \text{in } \Omega,$$
$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega,$$

where $\boldsymbol{f}$ is the vector of body forces, including the gravitational force, and the non-inertial terms

$$\boldsymbol{f} = \boldsymbol{g} - \boldsymbol{a} - \boldsymbol{\omega}\times(\boldsymbol{\omega}\times\boldsymbol{x}) - \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}t}\times\boldsymbol{x}$$

and $\boldsymbol{\varepsilon}(\boldsymbol{u})$ is the rate of deformation tensor given by

$$\boldsymbol{\varepsilon}(\boldsymbol{u}) = \tfrac{1}{2}(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^t).$$

Obviously, if $\boldsymbol{\omega}$ is time dependent, so will be the velocity and the pressure, and the local acceleration $\partial_t\boldsymbol{u}$ will have to be taken into account in the Navier–Stokes equations. However, for our purposes it is sufficient for

the moment to consider the stationary problem. The transient case will be discussed later on. The Navier–Stokes equations must be supplied with boundary conditions. We consider here the following two conditions of Dirichlet and Neumann types:

$$\begin{aligned} &\boldsymbol{u} = \boldsymbol{u}_g \quad \text{on } \Gamma_D, \\ &\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{t}_n \quad \text{on } \Gamma_N, \end{aligned} \tag{1}$$

where $\partial\Omega = \Gamma_N \cup \Gamma_D$, $\boldsymbol{n}$ is the outward unit normal to $\partial\Omega$, and $\boldsymbol{\sigma}$ is the stress tensor given by

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\nu\boldsymbol{\varepsilon}(\boldsymbol{u}),$$

$\boldsymbol{I}$ being the n_{d}-dimensional identity. We have chosen as Neumann condition the prescription of the traction $\boldsymbol{\sigma}\cdot\boldsymbol{n}$ because it usually enters naturally the variational form of the problem. However, we will also consider a particular weak form for which the natural condition not only involves the traction. Note that a mixed type of boundary prescriptions can also be considered. For example, in the numerical simulation of turbulent flows, it is common to consider an impermeable wall condition ($\boldsymbol{u}\cdot\boldsymbol{n} = 0$) together with the prescription of the tangential components of the traction to emulate the frictional effects of turbulent boundary layers [26]. To simplify the exposition, such boundary conditions will not be considered. Likewise, only laminar flows will be considered, although all what follows could be extended to include turbulence modeling.

Before going on to the variational formulation, we introduce the linearized Navier–Stokes operator $\boldsymbol{L}$ such that

$$\boldsymbol{L}(\boldsymbol{u},p) := \begin{bmatrix} (\overline{\boldsymbol{u}}\cdot\nabla)\boldsymbol{u} + 2\boldsymbol{\omega}\times\boldsymbol{u} - 2\nu\nabla\cdot\boldsymbol{\varepsilon}(\boldsymbol{u}) + \nabla p \\ \nabla\cdot\boldsymbol{u} \end{bmatrix}, \tag{2}$$

with $\overline{\boldsymbol{u}} = \boldsymbol{u}$ in the non-linear problem.

We now derive the variational formulation of our problem. Let us introduce the following functional spaces:

$$\begin{aligned} V &= \{\boldsymbol{v} \in H^1(\Omega)^{n_{\mathrm{d}}} | \boldsymbol{v}_{|\Gamma_D} = \boldsymbol{0}\}, \\ Q &= L^2(\Omega), \\ U &= \{\boldsymbol{v} \in H^1(\Omega)^{n_{\mathrm{d}}} | \boldsymbol{v}_{|\Gamma_D} = \boldsymbol{u}_g\}, \\ P &= \left\{ p \in L^2(\Omega) \middle| \int_\Omega p\,\mathrm{d}\Omega = 0 \quad \text{if } \Gamma_N = \emptyset \right\}. \end{aligned}$$

The first step to solve the Navier–Stokes equations is to linearize them. Let us denote denote by m the iteration number of the iterative scheme. For the sake of clarity, we only consider here the Picard linearization, that is,

$$[(\boldsymbol{u}\cdot\nabla)\boldsymbol{u}]^{m+1} \approx (\boldsymbol{u}^m\cdot\nabla)\boldsymbol{u}^{m+1}.$$

We are going to consider two weak forms of the Navier–Stokes equations. The first one is obtained by integrating by parts only the viscous term and is referred to as 0-weak formulation. The second one is obtained integrating by parts the viscous term and half of the convective term and is referred to as 1/2-weak formulation. We define a constant $\flat$ which can take the following values:

$$\begin{aligned} &\text{0-weak formulation:} \quad \flat = 0, \\ &\text{1/2-weak formulation:} \quad \flat = 1/2. \end{aligned}$$

The variational formulation of the problem reads as follows. Given $\boldsymbol{u}^0 \in U$, for $m = 0, 1, \ldots$ until convergence, find $(\boldsymbol{u}^{m+1}, p^{m+1}) \in U \times P$ such that

$$a^m(\boldsymbol{u}^{m+1},\boldsymbol{v}) - b(p^{m+1},\boldsymbol{v}) + b(q,\boldsymbol{u}^{m+1}) = l(\boldsymbol{v}) \tag{3}$$

for all $(\boldsymbol{v}\times q)\in V\times Q$, where

$$a^m(\boldsymbol{u},\boldsymbol{v}) = 2\int_\Omega \nu\boldsymbol{\varepsilon}(\boldsymbol{u}):\boldsymbol{\varepsilon}(\boldsymbol{v})\,\mathrm{d}\Omega + 2\int_\Omega(\boldsymbol{\omega}\times\boldsymbol{u})\cdot\boldsymbol{v}\,\mathrm{d}\Omega + (1-\flat)\int_\Omega[(\boldsymbol{u}^m\cdot\nabla)\boldsymbol{u}]\cdot\boldsymbol{v}\,\mathrm{d}\Omega$$
$$-\flat\int_\Omega[(\boldsymbol{u}^m\cdot\nabla)\boldsymbol{v}]\cdot\boldsymbol{u}\,\mathrm{d}\Omega + \flat\int_{\Gamma_N}(\boldsymbol{u}^m\cdot\boldsymbol{n})\boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}\Gamma,$$

$$b(p,\boldsymbol{v}) = \int_\Omega p\nabla\cdot\boldsymbol{v}\,\mathrm{d}\Omega,$$

$$l(\boldsymbol{v}) = \int_\Omega \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}\Omega + \int_{\Gamma_N}\boldsymbol{t}_n\cdot\boldsymbol{v}\,\mathrm{d}\Gamma.$$

2.2. Domain decomposition method

To explain our basic DD strategy we can consider a simple setting. Let us perform a geometrical decomposition of the original domain Ω into three disjoint and connected subdomains Ω_3, Ω_4 and Ω_5 such that

$$\Omega = \mathrm{int}(\overline{\Omega_3\cup\Omega_4\cup\Omega_5}).$$

From this partition, we define Ω_1 and Ω_2 as two overlapping subdomains:

$$\Omega_1 := \mathrm{int}(\overline{\Omega_3\cup\Omega_4}), \qquad \Omega_2 := \mathrm{int}(\overline{\Omega_5\cup\Omega_4}).$$

Finally, we define Γ_a as the part of $\partial\Omega_2$ lying in Ω_1, and Γ_b as the part of $\partial\Omega_1$ lying in Ω_2, formally given by

$$\Gamma_a := \overline{\partial\Omega_2\cap\Omega_1}, \qquad \Gamma_b := \overline{\partial\Omega_1\cap\Omega_2}.$$

The geometrical nomenclature is shown in Fig. 2. Γ_b and Γ_a are the *interfaces* of the domain decomposition method we now present. Ω_4 is the overlap zone. In the following, index i or j refers to a subdomain or an interface.

Let us denote $\Lambda_a := H_{00}^{1/2}(\Gamma_a)$. From the trace theorem (see e.g. [27,28]), we know that there exists a unique linear continuous map T_a, called the trace operator restricted to Γ_a, such that:

$$T_a : H^1(\Omega)^{n_\mathrm{d}} \to \Lambda_a^{n_\mathrm{d}}, \quad T_a\boldsymbol{v} = \boldsymbol{v}_{|\Gamma_a} \quad \forall\boldsymbol{v}\in H^1(\Omega)^{n_\mathrm{d}}.$$

In order to decouple the computation of the solution in the subdomains Ω_1 and Ω_2, we introduce a domain decomposition method. As we will show, this DD method is based on a Dirichlet/Neumann(Robin) coupling applied to overlapping subdomains. Let us define for $i=1,2$, $\Gamma_{D_i} := \Gamma_D\cap\partial\Omega_i$ and $\Gamma_{N_i} := \Gamma_N\cap\partial\Omega_i$. Let us introduce the following functional spaces:

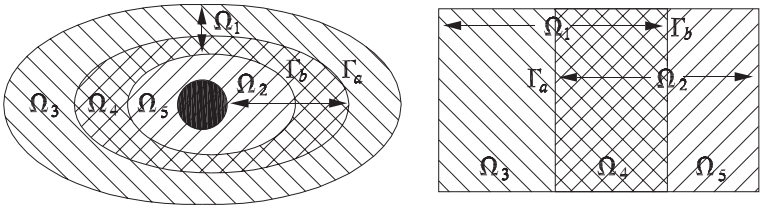

Fig. 2. Examples of decomposition of a domain Ω into two overlapping subdomains Ω_1 and Ω_2.

$$
\begin{aligned}
V_i &= \{\boldsymbol{v} \in H^1(\Omega_i)^{n_\mathrm{d}} | \boldsymbol{v}_{|\Gamma_{D_i}} = \boldsymbol{0}\},\\
V_i^0 &= \{\boldsymbol{v} \in H^1(\Omega_i)^{n_\mathrm{d}} | \boldsymbol{v}_{|\partial\Omega_i} = \boldsymbol{0}\},\\
Q_i &= L^2(\Omega_i),\\
U_i &= \{\boldsymbol{v} \in H^1(\Omega_i)^{n_\mathrm{d}} | \boldsymbol{v}_{|\Gamma_{D_i}} = \boldsymbol{u}_g\},\\
P_i &= \left\{ p \in L^2(\Omega_i) \middle| \int_{\Omega_i} p \,\mathrm{d}\Omega = 0 \quad \text{if } \Gamma_{N_i} = \emptyset \right\},\\
P_2^0 &= L^2(\Omega_2).
\end{aligned}
$$

To simplify the notation, let us omit the iteration superscript, knowing that the advection velocity is computed from the previous iteration. The algorithm we propose reads as follows: find $(\boldsymbol{u}_1, p_1) \in V_1 \times P_1$ and $(\boldsymbol{u}_2, p_2) \in V_2 \times P_2^0$ such that

$$
\begin{cases}
a_1(\boldsymbol{u}_1, \boldsymbol{v}_1) + b_1(q_1, \boldsymbol{u}_1) - b_1(p_1, \boldsymbol{v}_1) = l_1(\boldsymbol{v}_1) \quad \forall (\boldsymbol{v}_1, q_1) \in V_1^0 \times P_1,\\
\boldsymbol{u}_1 = \boldsymbol{u}_2 \quad \text{on } \Gamma_b,\\
a_2(\boldsymbol{u}_2, \boldsymbol{v}_2) + b_2(q_2, \boldsymbol{u}_2) - b_2(p_2, \boldsymbol{v}_2) = l_2(\boldsymbol{v}_2) \quad \forall (\boldsymbol{v}_2, q_2) \in V_2^0 \times P_2,\\
a_2(\boldsymbol{u}_2, \boldsymbol{E}_2\boldsymbol{\mu}_a) + a_3(\boldsymbol{u}_1, \boldsymbol{E}_3\boldsymbol{\mu}_a) = l_2(\boldsymbol{E}_2\boldsymbol{\mu}_a) + l_3(\boldsymbol{E}_3\boldsymbol{\mu}_a) + b_2(p_2, \boldsymbol{E}_2\boldsymbol{\mu}_a) + b_3(p_1, \boldsymbol{E}_3\boldsymbol{\mu}_a) \quad \forall \boldsymbol{\mu}_a \in \Lambda_a^{n_\mathrm{d}},
\end{cases}
\tag{4}
$$

where $\boldsymbol{E}_i$ denotes any possible extension operator

$$
\begin{aligned}
&\boldsymbol{E}_i : \Lambda_a^{n_\mathrm{d}} \to H^1(\Omega_i)^{n_\mathrm{d}},\\
&T_a \boldsymbol{E}_i \boldsymbol{\mu}_a = \boldsymbol{\mu}_a \quad \forall \boldsymbol{\mu}_a \in \Lambda_a^{n_\mathrm{d}}
\end{aligned}
$$

and a_i, b_i, l_i are the forms introduced before with the integrals extended over Ω_i. Under some conditions, it can be shown that this formulation is equivalent to the original stationary counterpart of Eq. (3). This holds for the discrete case as well. See for example [29] for the Stokes problem with *disjoint* subdomains and [30] for advection–diffusion problems with overlapping subdomains. However, the form of the system of equations $(4)_{1\text{–}4}$ is not convenient for our objective. To change it, we can make use of the fact that *the solution of the domain decomposition problem satisfies*

$$
\boldsymbol{\sigma}_1 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_1 \cdot \boldsymbol{n}_2)\boldsymbol{u}_1 = \boldsymbol{\sigma}_2 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_2 \cdot \boldsymbol{n}_2)\boldsymbol{u}_2 \quad \text{on } \Gamma_a, \text{ in the sense of } \Lambda_a,
$$

where $\partial(\cdot)/\partial n_2 = \boldsymbol{n}_2 \cdot \nabla(\cdot)$, $\boldsymbol{n}_2$ being the outward unit vector normal to Ω_2 on the interface Γ_a.

Let us prove this result. Note first that according to Green's formula, we have for all $\boldsymbol{\mu}_a \in \Lambda_a^{n_\mathrm{d}}$

$$
a_3(\boldsymbol{u}_1, \boldsymbol{E}_3\boldsymbol{\mu}_a) - b_3(p_1, \boldsymbol{E}_3\boldsymbol{\mu}_a) = -\langle \boldsymbol{\sigma}_1 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_1 \cdot \boldsymbol{n}_2)\boldsymbol{u}_1, \boldsymbol{\mu}_a \rangle_{\Gamma_a} + \langle \boldsymbol{\sigma}_1 \cdot \boldsymbol{n}, \boldsymbol{E}_3\boldsymbol{\mu}_a \rangle_{\Gamma_{N_1}} + \langle L_1(\boldsymbol{u}_1, p_1), \boldsymbol{E}_3\boldsymbol{\mu}_a \rangle_{\Omega_3},
\tag{5}
$$

$$
a_2(\boldsymbol{u}_2, \boldsymbol{E}_2\boldsymbol{\mu}_a) - b_2(p_2, \boldsymbol{E}_2\boldsymbol{\mu}_a) = \langle \boldsymbol{\sigma}_2 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_2 \cdot \boldsymbol{n}_2)\boldsymbol{u}_2, \boldsymbol{\mu}_a \rangle_{\Gamma_a} + \langle \boldsymbol{\sigma}_2 \cdot \boldsymbol{n}, \boldsymbol{E}_2\boldsymbol{\mu}_a \rangle_{\Gamma_{N_2}} + \langle L_1(\boldsymbol{u}_2, p_2), \boldsymbol{E}_2\boldsymbol{\mu}_a \rangle_{\Omega_2},
\tag{6}
$$

where $\langle \cdot, \cdot \rangle_\omega$ denotes the duality pairing in $H^1 \times (H^1)'$ if ω is a n_d-dimensional domain and in $H_{00}^{1/2}(\Gamma_a) \times H^{-1/2}(\Gamma_a)$ if its dimension is $n_\mathrm{d} - 1$. The operator L_1 is the first row of $\boldsymbol{L}$ in Eq. (2) with $\overline{\boldsymbol{u}} = \boldsymbol{u}$. In addition, from Eqs. $(4)_1$ and $(4)_3$, we have, for $q_1 = 0$ and $q_2 = 0$,

$$
L_1(\boldsymbol{u}_1, p_1) = \boldsymbol{f} \quad \text{in } \Omega_1,
\tag{7}
$$

$$
L_1(\boldsymbol{u}_2, p_2) = \boldsymbol{f} \quad \text{in } \Omega_2,
\tag{8}
$$

in the sense of distributions. As a result, and using Eq. (1) for the definition of the traction on Γ_N, Eqs. (5) and (6) become

$$
a_3(\boldsymbol{u}_1, \boldsymbol{E}_3\boldsymbol{\mu}_a) - b_3(p_1, \boldsymbol{E}_3\boldsymbol{\mu}_a) = -\langle \boldsymbol{\sigma}_1 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_1 \cdot \boldsymbol{n}_2)\boldsymbol{u}_1, \boldsymbol{\mu}_a \rangle_{\Gamma_a} + \langle \boldsymbol{t}_n, \boldsymbol{E}_3\boldsymbol{\mu}_a \rangle_{\Gamma_{N_1}} + \langle \boldsymbol{f}, \boldsymbol{E}_3\boldsymbol{\mu}_a \rangle_{\Omega_3},
$$

$$a_2(\boldsymbol{u}_2, \boldsymbol{E}_2\boldsymbol{\mu}_a) - b_2(p_2, \boldsymbol{E}_2\boldsymbol{\mu}_a) = \langle \boldsymbol{\sigma}_2 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_2 \cdot \boldsymbol{n}_2)\boldsymbol{u}_2, \boldsymbol{\mu}_a \rangle_{\Gamma_a} + \langle \boldsymbol{t}_n, \boldsymbol{E}_2\boldsymbol{\mu}_a \rangle_{\Gamma_{N_2}} + \langle \boldsymbol{f}, \boldsymbol{E}_2\boldsymbol{\mu}_a \rangle_{\Omega_2}.$$

Adding up the two equations, and substituting the result into Eq. $(4)_4$, we find

$$\langle -\boldsymbol{\sigma}_1 \cdot \boldsymbol{n}_2 + \flat(\boldsymbol{u}_1 \cdot \boldsymbol{n}_2)\boldsymbol{u}_1 + \boldsymbol{\sigma}_2 \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{u}_2 \cdot \boldsymbol{n}_2)\boldsymbol{u}_2, \boldsymbol{\mu}_a \rangle_{\Gamma_a} = 0$$

and thus the claim is proved.

Eq. $(4)_4$ represents therefore the continuity of the natural condition across the interface. In the case of the 0-weak form, this condition is simply the traction; in the case of the 1/2-form, this is a more general Robin condition.

2.3. Iteration-by-subdomain method

The DD method presented previously consists in solving two separate subproblems in Ω_1 and Ω_2 and couple them to obtain a global solution on the original computational domain. The coupling is obtained by imposing the continuity of the Dirichlet data on one interface and the continuity of the Neumann (or Robin) data, appearing as a natural boundary condition, on the other interface. We now set up an iteration-by-subdomain algorithm in order to decouple the solution of the subproblems. Suppose that we have the solution $(\boldsymbol{u}^m, p^m)$ at the mth iteration step of the iterative scheme to deal with the non-linearity of the problem, and we want to compute $(\boldsymbol{u}^{m+1}, p^{m+1})$. We propose to obtain this pair through a nested iterative scheme, whose iteration counter is denoted by k. If $\boldsymbol{\sigma}_i^{m+1,k}$ is the stress computed from $(\boldsymbol{u}_i^{m+1,k}, p_i^{m+1,k})$, for $i = 1, 2$ corresponding to the subdomains Ω_i, the iteration-by-subdomain algorithm reads as follows: given $\boldsymbol{u}_1^{m+1,0} = \boldsymbol{u}_1^m \in U_1$ and $\boldsymbol{u}_2^{m+1,0} = \boldsymbol{u}_2^m \in U_2$, for each $m \geqslant 0$, find until convergence $(\boldsymbol{u}_1^{m+1,k+1}, p_1^{m+1,k+1}) \in U_1 \times P_1$ and $(\boldsymbol{u}_2^{m+1,k+1}, p_2^{m+1,k+1}) \in U_2 \times P_2^0$, for $k = 0, 1, 2, \ldots$, such that

$$\begin{cases} a_1^m(\boldsymbol{u}_1^{m+1,k+1}, \boldsymbol{v}_1) + b_1(q_1, \boldsymbol{u}_1^{m+1,k+1}) - b_1(p_1^{m+1,k+1}, \boldsymbol{v}_1) = l_1(\boldsymbol{v}_1) \quad \forall (\boldsymbol{v}_1, q_1) \in V_1^0 \times P_1, \\ \boldsymbol{u}_1^{m+1,k+1} = \boldsymbol{u}_2^{m+1,k} \quad \text{on } \Gamma_b, \\ a_2^m(\boldsymbol{u}_2^{m+1,k+1}, \boldsymbol{v}_2) + b_2(q_2, \boldsymbol{u}_2^{m+1,k+1}) - b_2(p_2^{m+1,k+1}, \boldsymbol{v}_2) \\ \quad = l_2(\boldsymbol{v}_2) + \langle \boldsymbol{\sigma}_1^{m+1,k+1} \cdot \boldsymbol{n} - \flat(\boldsymbol{u}_1^{m+1,k+1} \cdot \boldsymbol{n})\boldsymbol{u}_1^{m+1,k+1}, \boldsymbol{v}_2 \rangle_{\Gamma_a} \quad \forall (\boldsymbol{v}_2, q_2) \in V_2 \times P_2. \end{cases} \tag{9}$$

This iterative procedure is a Dirichlet/Neumann or Dirichlet/Robin iteration-by-subdomain method on overlapping subdomains if we use the 0-weak form or the 1/2-weak form, respectively. In this iterative procedure, we have chosen to nest the linearization and DD iterative loops, the inner loop being the DD one. Letting $\boldsymbol{u}^m = \boldsymbol{u}^{m+1,k}$ in the convective term, we couple both loops. In the discrete finite element problem, it is computationally preferable to choose the DD loop as the inner loop, as, on the contrary, the matrix of the discrete system would have to be computed at each iteration.

Note that if $\Gamma_{N_1} = \emptyset$, then we must ensure to satisfy the compatibility condition in subdomain 1 which requires that

$$\int_{\partial\Omega_1} \boldsymbol{u}_1^{m+1,k+1} \cdot \boldsymbol{n} \, \mathrm{d}\Gamma = 0.$$

The initial Dirichlet transmission condition $\boldsymbol{u}_2^{0,0}$ on Γ_b must therefore be chosen such that we have zero flux on $\partial\Omega_1$ at the first iteration. If this condition is satisfied, then it will be satisfied at further iterations, since for the continuous problem mass conservation is verified pointwise.

Finally, let us note that we can introduce relaxation parameters to update the transmission conditions. We will denote θ_{D} and θ_{R} the relaxation factors of the Dirichlet and Robin transmission conditions, respectively.

2.4. Summary of results for advection–diffusion problems

In [30], we introduced a Dirichlet/Robin iteration-by-subdomain method for the solution of advection–diffusion-reaction problems on overlapping subdomains. For this linear scalar problem we were able to perform a complete numerical analysis. The main results we proved are:

- The DD is well defined, that is, the solution of the decomposed problem using Dirichlet/Neumann (Robin) transmission conditions with *overlapping* subdomains is *the same* as for the original problem.
- The unknowns at the interfaces are the unique solutions of problems involving generalizations of the classical Steklov–Poincaré operators.
- The iteration-by-subdomain scheme converges linearly, provided the under-relaxation parameters are sufficiently small.

Likewise, in [31], we considered the differential equivalent of the iteration-by-subdomain method and tested it for a one-dimensional problem. An overlapping Dirichlet/Neumann method is also considered. The main conclusions drawn in this case are:

- The Dirichlet/Robin method tends to the Schwarz method when the diffusion tends to zero, while it behaves like the Dirichlet/Neumann method in the diffusion-dominated range.
- The overlapping makes the Dirichlet/Neumann and Dirichlet/Robin methods more robust.
- In the hyperbolic limit, the overlap enables to diffuse the error much more rapidly than the D/N and D/R methods do on disjoint subdomains.
- In the hyperbolic limit, it is well known that the standard Dirichlet/Neumann method does not converge when the Neumann interface is placed at the inflow. In the presence of reaction, and as long as the overlap is greater than the artificial internal layer created at the interfaces, the Dirichlet/Neumann converges independently of whether the transmission condition is placed at the inflow or not. Note that when using a finite difference scheme to discretize the temporal derivative in the case of a transient problem, the additional term introduced in the reaction term behaves, in many respects, like the inflow BC. The method will be more robust when applied to transient problems than stationary ones.

We expect these properties to be inherited by the present domain decomposition method applied to the Navier–Stokes equations. However, a rigorous theoretical analysis of this problem is still open.

3. Discretization and implementation aspects

3.1. Discretization in time and space

We now consider the transient Navier–Stokes equations:

$$\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + 2\boldsymbol{\omega}\times\boldsymbol{u} - 2\nu\nabla\cdot\boldsymbol{\varepsilon}(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \quad \text{in } \Omega\times(0,T),$$

$$\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega\times(0,T),$$

$$\boldsymbol{u} = \boldsymbol{u}_g \quad \text{on } \Gamma_D\times(0,T),$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n} = \boldsymbol{t}_n \quad \text{on } \Gamma_N\times(0,T),$$

$$\boldsymbol{u} = \boldsymbol{u}^0 \quad \text{on } \Omega\times\{0\},$$

where the force term and the boundary conditions can depend on time. The time interval where the problem is to be solved is $(0,T)$. The temporal derivative of the velocity has been denoted by $\partial_t \boldsymbol{u}$, t being the time variable.

The time discretization is carried out using the generalized trapezoidal rule, i.e. a finite difference scheme. Let us introduce a uniform partition of the time interval $[0,T]$ and define, for $\theta \in (0,1]$,

$$\boldsymbol{u}^{n+\theta} := \theta \boldsymbol{u}^{n+1} + (1-\theta)\boldsymbol{u}^n, \quad \delta t := t^n - t^{n-1}, \quad \delta_t \boldsymbol{u}^{n+\theta} := \frac{\boldsymbol{u}^{n+\theta} - \boldsymbol{u}^n}{\theta\, \delta t},$$

where δt is the time step size and superscript n denotes the approximated solution at time $n\,\delta t$. According to this integration rule, the time-discretized Navier–Stokes equations are solved as follows. Given an initial condition $\boldsymbol{u}^0$, find $\boldsymbol{u}^{n+1}$ and p^{n+1} for each $n \geqslant 0$ such that

$$\delta_t \boldsymbol{u}^{n+\theta} + (\boldsymbol{u}^{n+\theta}\cdot\nabla)\boldsymbol{u}^{n+\theta} + 2\boldsymbol{\omega}^{n+\theta}\times\boldsymbol{u}^{n+\theta} - 2\nu\nabla\cdot\boldsymbol{\varepsilon}(\boldsymbol{u}^{n+\theta}) + \nabla p^{n+\theta} = \boldsymbol{f}^{n+\theta},$$
$$\nabla\cdot\boldsymbol{u}^{n+\theta} = 0,$$

in Ω, with the following boundary conditions

$$\boldsymbol{u}^{n+\theta} = \boldsymbol{u}_g \quad \text{on } \Gamma_D \text{ at time } t^n + \theta\,\delta t,$$
$$\boldsymbol{\sigma}^{n+\theta}\cdot\boldsymbol{n} = \boldsymbol{t}_n \quad \text{on } \Gamma_N \text{ at time } t^n + \theta\,\delta t,$$

where $\boldsymbol{\sigma}^{n+\theta} = -p^{n+\theta}\boldsymbol{I} + 2\nu\boldsymbol{\varepsilon}(\boldsymbol{u}^{n+\theta})$. The unknown at time step $n+1$ is obtained using the fact that $x^{n+1} = x^n + (x^{n+\theta} - x^n)/\theta$ for each unknown x of the problem. The choice $\theta = 1$ corresponds to the backward Euler scheme, unconditionally stable and of first order. The choice $\theta = 0.5$ corresponds to the Crank–Nicholson scheme, also unconditionally stable but of second order.

Let us consider now the space discretization. Let $\{\Omega^e\}$ be a regular partition of the domain Ω, with index e ranging from 1 to the number of elements n_e. The finite element approximation we will consider is conforming, that is, the discrete spaces of test functions and of trial solutions will be linear subspaces of the corresponding spaces for the continuous problem, associated to the partition $\{\Omega^e\}$. We will denote them by a superscript h.

We denote by $x^{n+\theta,m+1}$ the variable x considered at linearization step $m+1$ and time level $n+\theta$. The Galerkin formulation of the problem reads as follows. Given $\boldsymbol{u}_h^{n+\theta,0} = \boldsymbol{u}_h^n \in U_h$, for each time step $n \geqslant 0$, find for $m = 0, 1, \ldots$ until convergence, $(\boldsymbol{u}_h^{n+\theta,m+1}, p_h^{n+\theta,m+1}) \in U_h \times P_h$ such that

$$(\delta_t \boldsymbol{u}_h^{n+\theta,m+1}, \boldsymbol{v}_h) + a^{n+\theta,m}(\boldsymbol{u}_h^{n+\theta,m+1}, \boldsymbol{v}_h) - b(p_h^{n+\theta,m+1}, \boldsymbol{v}_h) + b(q_h, \boldsymbol{u}_h^{n+\theta,m+1}) = l^{n+\theta}(\boldsymbol{v}_h),$$

for all $(\boldsymbol{v}_h \times q_h) \in V_h \times Q_h$, where the superscript $n+\theta$ in l denotes that the external forces are evaluated at this time level, whereas the superscript $n+\theta, m$ in the bilinear form a indicates that it is computed with the advection velocity $\boldsymbol{u}_h^{n+\theta,m}$.

It is well known that the Galerkin formulation can lack stability for three major reasons. The first reason is related to the compatibility of the finite element spaces for the velocity and the pressure which have to satisfy the so-called Ladyzhenskaya–Brezzi–Babuška condition [32]. This condition is necessary to obtain a stability estimate for the pressure; without requiring this condition, the pressure would be out of control. The second reason is attributed to the relative importance of the viscous and convective effects. Finally, the third one appears when the Coriolis force becomes important with respect to viscous effects. We will now present a stabilized formulation, based on the algebraic variational subgrid scale (SGS) model first introduced in [33]. The variational SGS model uses as a starting argument that the lack of resolution achieved by the mesh is responsible for the numerical instabilities. Therefore, the model calculates in some approximate way the unresolved scales of the flow, i.e. the scales smaller than the mesh size. The method is extensively described in [34] and [35].

After time discretization and linearization, the problem can be re-written in a compact form as

$$[\delta_t \boldsymbol{u}^{n+\theta,m+1}, 0]^t + \boldsymbol{L}^{n+\theta,m}(\boldsymbol{u}^{n+\theta,m+1}, p^{n+\theta,m+1}) = \boldsymbol{F}^{n+\theta} \quad \text{in } \Omega,$$

where $\boldsymbol{L}^{n+\theta,m}$ is defined as

$$\boldsymbol{L}^{n+\theta,m}(\boldsymbol{u}, p) := \begin{bmatrix} (\boldsymbol{u}^{n+\theta,m} \cdot \nabla)\boldsymbol{u} + 2\boldsymbol{\omega}^{n+\theta} \times \boldsymbol{u} - 2\nu\nabla \cdot \boldsymbol{\varepsilon}(\boldsymbol{u}) + \nabla p \\ \nabla \cdot \boldsymbol{u} \end{bmatrix}$$

and the force term is defined as

$$\boldsymbol{F}^{n+\theta} := \begin{bmatrix} \boldsymbol{f}^{n+\theta} \\ 0 \end{bmatrix}.$$

We finally define the residual of the Navier–Stokes equations $\boldsymbol{R}^{n+\theta,m+1}$ at iteration $m+1$ of the time step $n+\theta$ as

$$\boldsymbol{R}^{n+\theta,m+1}(\boldsymbol{u}_h^{n+\theta,m+1}, p_h^{n+\theta,m+1}) := [\delta_t \boldsymbol{u}_h^{n+\theta,m+1}, 0]^t + \boldsymbol{L}^{n+\theta,m}(\boldsymbol{u}_h^{n+\theta,m+1}, p_h^{n+\theta,m+1}) - \boldsymbol{F}^{n+\theta}.$$

The viscous term in the right-hand side can be evaluated in the interior of the elements. The stabilized weak form reads: given $\boldsymbol{u}_h^{n+1,0} = \boldsymbol{u}_h^n \in U_h$, for each time step $n \geqslant 0$, find for $m = 0, 1, \ldots$ until convergence, $(\boldsymbol{u}_h^{n+1,m+1}, p_h^{n+1,m+1}) \in U_h \times P_h$ such that

$$(\delta_t \boldsymbol{u}_h^{n+\theta,m+1}, \boldsymbol{v}_h) + a^{n+\theta,m}(\boldsymbol{u}_h^{n+\theta,m+1}, \boldsymbol{v}_h) - b(p_h^{n+\theta,m+1}, \boldsymbol{v}_h) + b(q_h, \boldsymbol{u}_h^{n+\theta,m+1}) - \sum_{e=1}^{n_e} \int_{\Omega_e} \boldsymbol{L}^{*n+\theta,m}(\boldsymbol{v}_h, q_h)^t \boldsymbol{\tau}_e \boldsymbol{R}^{n+\theta,m+1}(\boldsymbol{u}_h^{n+\theta,m+1}, p_h^{n+\theta,m+1}) = l^{n+\theta}(\boldsymbol{v}_h),$$

$\forall (\boldsymbol{v}_h, q_h) \in V_h \times Q_h$, where $\boldsymbol{L}^{*n+\theta}$ is the adjoint of $\boldsymbol{L}^{n+\theta}$, given by

$$\boldsymbol{L}^{*n+\theta,m}(\boldsymbol{v}, q) := \begin{bmatrix} -(\boldsymbol{u}^{n+\theta,m} \cdot \nabla)\boldsymbol{v} - 2\boldsymbol{\omega}^{n+\theta} \times \boldsymbol{v} - 2\nu\nabla \cdot \boldsymbol{\varepsilon}(\boldsymbol{v}) - \nabla q \\ -\nabla \cdot \boldsymbol{v} \end{bmatrix}.$$

$\boldsymbol{\tau}_e$ is the matrix of stabilization parameters, computed in each element as [34]

$$\boldsymbol{\tau}_e = \operatorname{diag}(\tau_1 \boldsymbol{I}, \tau_2), \quad \text{where}$$

$$\tau_1 = \left(\frac{c_1 \nu}{h_e^2} + \frac{c_2 |\boldsymbol{u}^{n+\theta,m}|}{h_e} + c_3 |\boldsymbol{\omega}^{n+\theta}| \right)^{-1},$$

$$\tau_2 = c_4 \frac{h_e^2}{\tau_1}.$$

τ_2 contributes to enforcing the incompressibility of the flow, which is excessively relaxed by the term multiplied by τ_1. The values of the algorithmic constants we use are $c_1 = 4$, $c_2 = 2$, $c_3 = 1$, $c_4 = 1$ and h_e is the characteristic element length. For quadratic elements, h_e is taken as half of the element size.

Remark 1. The iteration-by-subdomain method given by Eqs. $(9)_{1-3}$ cannot be applied at the discrete level without special care. We note that to obtain the iterative scheme $(9)_{1-3}$ from Eqs. $(4)_{1-4}$, we had to use Eqs. (7) and (8). But in the discrete case, these two equations are not satisfied pointwise (in general). Thus, while the original and the decomposed problem are equivalent at the continuous level (for smooth solutions), their discrete counterparts are not. Nevertheless, we will see in the following subsection that we have a way to keep the order of convergence in space of the algorithm using a simple scheme to calculate the derivatives involved in the transmission condition. Now, if we want to go on to the transient case, we just have to apply algorithm $(9)_{1-3}$ at each time step.

3.2. A Master/Slave-coupling algorithm

3.2.1. General algorithm

In this section we present an implementation algorithm for general iteration-by-subdomain methods. We denote the subdomains i or j, with transmission conditions of Dirichlet, Neumann and Robin types, involving overlapping and non-overlapping meshes. For the sake of clarity, we assume that the subdomains are steady (not the flows). The iterative domain decomposition algorithm consists of three steps: the pre-process, the process and the post-process.

- Pre-process: The pre-process consists in dividing the computational domain into overlapping and/or non-overlapping subdomains. The nodes involved in the transmission process are called the *interface nodes*.
- Process: The DD method we propose is algorithmic because the solution on each subdomain is obtained on separate processes and the exchange of information between the subdomains is carried out by a master code. Communication between the master code and the slave codes (Navier–Stokes solver on each subdomain) can be achieved by any of the communication libraries like PVM or MPI. Each subproblem runs on different processes of the Navier–Stokes solver. The master code controls the iterative process and has to perform the following operations:
  - find the host elements of the interface nodes in the adjacent subdomain;
  - interpolate the variables from one subdomain to another;
  - update of the boundary condition of each one;
  - pass the data (the new boundary condition) back and forth to the slaves (the processes of the finite element code).
- Post-process: Eventually, the post-process defines the global solution. For example, in the case of overlapping grids, one has to define the solution in the regions in common.

We now describe the specific tasks to be carried out by the Master and the Slaves. Within a standard implicit Navier–Stokes solver, the DD algorithm loop fits within a multi-loop algorithm as shown by Algorithm 1.

Algorithm 1. Slaves point of view

```
for time steps do
  for linearization steps do
    for DD steps do
      Import transmission condition update from Master
      for solver steps do
        Solve Algebraic system
      end for
    end for
    Export new solution to Master
  end for
end for
```

With respect to a classical solver, as shown by Algorithm 1, the DD algorithm is just an additive loop that can be coupled with the others. For example, it may be convenient to couple it with the linearization loop, which is the strategy employed in all the examples presented in this work. When using explicit codes, the DD loop can also be coupled with the time loop.

Let us assume we want to couple n_s subdomains. We denote by Γ_{ij} the interface of subdomain i with subdomain j. The n_s slave processes are distributed via a multicoloring technique: each subdomain is

assigned a color color(i) so that subdomains of the same color have no common interface. The colors are ordered from 1 to n_c, where n_c is the total number of colors used. Subdomains of smaller color are run first.

The algorithm as seen from the master code is shown in Algorithm 2. The first task is to find the host elements of all the interface nodes to enable further interpolation of the transmission conditions. When the subdomains are steady, this operation must be performed only once, as a pre-process work. The search technique used in this work will be described in Section 3.2.2.

The stopping criterion is based on some norm of the interface unknown changes between two successive iterates, k and $k+1$. We define the *interface L^2 residual* of variable x as the following quantity

$$\left[\sum_{i=1}^{n_s}\sum_{j}\left(\frac{1}{|\Gamma_{ij}|}\int_{\Gamma_{ij}}(x_i^{k+1}-x_i^k)^2\,\mathrm{d}\Gamma\right)\right]^{1/2}\left[\sum_{i=1}^{n_s}\sum_{j}\left(\frac{1}{|\Gamma_{ij}|}\int_{\Gamma_{ij}}(x_i^{k+1})^2\,\mathrm{d}\Gamma\right)\right]^{-1/2}, \tag{10}$$

where $|\Gamma_{ij}|$ is the measure of Γ_{ij} and x_i^k is the approximate solution on Γ_{ij} obtained solving in subdomain i at iteration k. The sum in j is extended to all subdomains connected to i.

Algorithm 2. Master's point of view

```
Impose initial conditions
for time steps do
  Set iteration number k = 0
  Find the host elements of all the Γ_ij interface nodes of subdomain i in subdomain j
  while stopping criterion not reached do
    for color = 1 to n_c do
      Export transmission conditions to subdomains i such that color(i) = color
      Run subdomains i in parallel
      Import solutions from subdomains i
      for subdomains j connected to subdomain i do
        Interpolate and compute transmission conditions for subdomains j
      end for
    end for
    k = k + 1
    Check convergence of the DD scheme
  end while
end for
```

3.2.2. Search algorithm

Once the domain decomposition has been performed, one of the first tasks of the Master is to find host elements for all the interface nodes. The element search strategy (ESS) used in this work is based on a quad-tree strategy (in 2D, and oct-tree in 3D). The strategy can be decomposed in two steps, the pre-process (which constructs the tree-like structure) and the process (*range searching*). In the pre-process, the computational domain is first embedded in a box, taking the minimum and maximum of the node coordinates to define its corners. The algorithm recursively partitions the box(es) into smaller boxes, until each box contains less than a prescribed number of nodes. A box is divided into four boxes in two dimensions (quad-tree) and into eight boxes in three dimensions (oct-tree). If a box is not divided further because it contains too few nodes, we start filling it with elements. To get the list of elements located inside a box, the node/element connectivity is used; all the elements connected to the node of a box belong to this box. Note that this criterion does not require any calculation of the intersections of the faces of elements and boxes. Fig. 3 shows a two-dimensional example of a quad-tree subdivision of a mesh of a NACA0012.

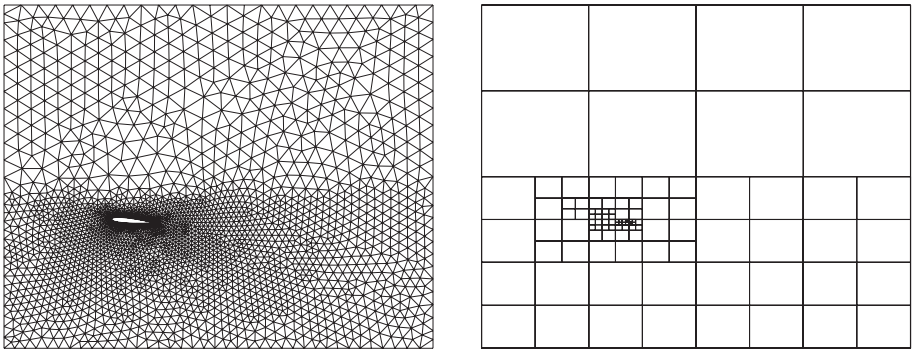

Fig. 3. A quad-tree division of a mesh (NACA0012). (Left) Mesh. (Right) Quad-tree structure.

The process step is known as range searching. Knowing the coordinates of the test point we proceed down through the tree until we find a box containing elements where the test point falls. Now we perform a loop over the elements belonging to the box, compute the natural coordinates of the point in each of these elements until we find the host element. The condition for an element to belong to a box (based on the connectivity) can seem restrictive; in fact, an element can intersect a box without having any node in it. Nevertheless, it has proved to be sufficient for most cases and the search almost never fails. In case a point has no host element, a new search is performed using a less restrictive method.

3.2.3. Interpolation of Dirichlet, Neumann and Robin data

Up to now we have studied how the transmission conditions are passed from one subdomain to another to set up the iterative algorithm. We now study the interpolation technique performed at the finite element level. In the following, we assume we want to update the solution of subdomain i knowing the solution of subdomain j. For the sake of clarity, we drop the superscripts referring to iterative loops as well as the subscript h referring to discrete solutions.

3.2.3.1. Interpolation of Dirichlet data. The Dirichlet data are all the velocity components and their interpolations are carried out as follows. From the element search strategy, we have identified the host elements `jelem` of each interface node `ipoin`, as well as the natural coordinates of `ipoin` in `jelem`; see Fig. 4 (left). The new value of the velocity at `ipoin` is simply obtained by interpolating the velocity using the classical Lagrange interpolation of element `jelem`.

Let us mention that this interpolation is "diffusive" as some information can be missed during the interpolation. For example, this may be the case when the mesh of subdomain i is coarser than the mesh of subdomain j; this point is known as *conservation* and will not be treated here. In addition if subdomain i is confined, we may violate the mass conservation in Ω_i as we have no control over the mass interpolated from Ω_j and passing through Γ_{ij}. This is not the case in the continuous problem where mass is conserved

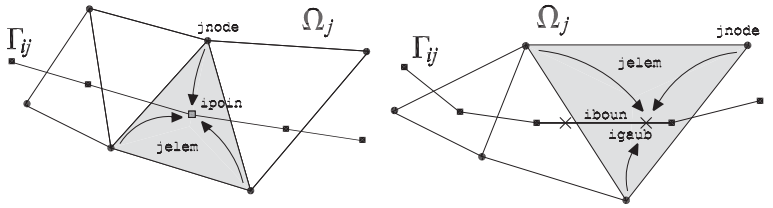

Fig. 4. Interpolation. (Left) Dirichlet data. (Right) Neumann and Robin data.

pointwise. The solution to these deficiencies we use is explained in [36]. The idea is that instead of taking the plain Lagrangian interpolation as transmission condition, we take the closest approximation (in the L^2 sense) that satisfies some a priori restrictions (such as mass conservation, for example).

3.2.3.2. Interpolation of Neumann and Robin data. The Neumann or Robin data are involved in the natural transmission conditions. In the last section, we mentioned in Remark 1 that the formulation of the DD method given by Eqs. $(9)_{1-3}$ could not be directly extended to the discrete case, the reason being that the original differential equations are not satisfied pointwise.

The Neumann (or Robin) transmission condition of formulation of the DD method (Eqs. $(9)_{1-3}$) consists in calculating the following contour integral on the interface Γ_{ij}:

$$\int_{\Gamma_{ij}} (\boldsymbol{\sigma}_j \cdot \boldsymbol{n} - \flat(\boldsymbol{u}_j \cdot \boldsymbol{n})\boldsymbol{u}_j) \cdot \boldsymbol{v}_i \, \mathrm{d}\Gamma = \int_{\Gamma_{ij}} (2\nu\boldsymbol{\varepsilon}(\boldsymbol{u}_j) \cdot \boldsymbol{n} - p_j\boldsymbol{n} - \flat(\boldsymbol{u}_j \cdot \boldsymbol{n})\boldsymbol{u}_j) \cdot \boldsymbol{v}_i \, \mathrm{d}\Gamma, \tag{11}$$

where $(\boldsymbol{u}_j, p_j)$ are known from the previous solution on Ω_j. We are interested here in the calculation of the first-order derivatives of the velocity components involved in the strain rate tensor.

If formulae (5) and (6) are applied to the discrete finite element problem, the second derivatives in L_1 must be understood in the sense of distributions. The δ-like terms associated to the edges interior to the subdomain considered contribute with the force term $\boldsymbol{f}$, whereas the terms associated to the boundary edges contribute to the normal component of the strain rate tensor in (11). The final result is that this term *has contributions from both sides of* Γ_{ij}, that is, from Ω_i and from $\Omega_j \setminus \Omega_i$.

In the case of *disjoint* subdomains one could apply a technique similar to those employed to deal with second-order terms in discontinuous Galerkin finite element methods (see e.g. [37]). However, it is not clear how to do this in the general setting we consider. We will explain now two possibilities to compute $\boldsymbol{\varepsilon}(\boldsymbol{u}_j)$ in Eq. (11), one based only on the values of the derivatives in the elements of Ω_j crossing Γ_{ij} and the other on the values of the derivatives in Ω_j obtained after a least-square smoothing, and thus involving a wider region of at least two layers of elements in Ω_j around Γ_{ij}, if the overlapping is wide enough. We will refer to both approaches as the one-layer and the multi-layer interpolations.

We start by presenting the one-layer interpolation scheme. We first note that the velocity derivatives are needed at the integration points of the boundaries in order to perform the numerical integration. Let us consider the element boundary `iboun` and define `igaub` as an integration point on this boundary; see Fig. 4 (right) for notation. Once the host element `jelem` of `igaub` in subdomain j is found, we obtain the one-layer interpolation by direct interpolation of the derivatives from the nodes (if available) or from the integration points of `jelem` to the boundary integration points. The strategy to compute the force term (11) is shown in Algorithm 3. Note that for disjoint subdomains, the derivatives will come only from one side of Γ_{ij}, which leads to a poor approximation.

Algorithm 3. One-layer interpolation

for all boundary elements `iboun` **do**
  for all integration points `igaub` **do**
    Find host element `jelem` of `igaub`
    Interpolate derivatives $\nabla\boldsymbol{u}_j$ from nodes `jnode` to `igaub`
    Calculate outward unit normal $\boldsymbol{n}$ and test function $\boldsymbol{v}_i$ at `igaub`
    Calculate the product $2\nu\boldsymbol{n} \cdot \boldsymbol{\varepsilon}(\boldsymbol{u}_j) \cdot \boldsymbol{v}_i$ at `igaub` and multiply the result by the weight of the numerical integration
    Assemble result
  end for
end for

The interpolation of the other two terms involved in the transmission condition (11) does not pose any particular difficulty, and we just discuss briefly the interpolation of the pressure. On the one hand, when using continuous pressure spaces, the pressure is interpolated at the boundary integration points in a classical way, i.e. like the velocity. On the other hand, when using discontinuous pressure spaces, the pressure is first smoothed using the least-squares smoothing, before being interpolated from the nodes to the boundary integration points.

Let us consider now the multi-layer approach. We explained previously that the interpolations of the velocity derivatives need information from the background mesh on both sides of the interface. One way of applying this is to perform a least-squares smoothing to compute the derivatives of the unknown at the nodes of subdomain j. By doing so, the values of the derivatives at a node `jnode` will depend on the derivatives calculated on all its neighboring elements, i.e. on the values of the function at all the nodes of the elements connected to `jnode`. We showed in [31] that we could obtain a second-order convergence in one dimension for linear elements; we expect that the method to be presented here for two and three dimensions will maintain the expected space accuracy as well.

The least-square smoothing used here is standard; see for example [38]. The matrix system resulting from the least-square smoothing involves a mass matrix. This system can be solved efficiently using a closed quadrature rule, for which the integration points are located at the nodes. The associated mass matrix is diagonal and therefore it can be trivially inverted.

Once the derivatives are obtained at the nodes of the background mesh, we proceed as in the case of the one-layer interpolation, as shown in Algorithm 4. The (projected) velocity gradient obtained after smoothing in Ω_j has been denoted $\Pi_h(\nabla \boldsymbol{u}_j)$.

Algorithm 4. Multi-layer interpolation
Perform least-squares smoothing for the derivatives
for all boundary elements `iboun` **do**
  for all integration points `igaub` **do**
    Find host element `jelem` of `igaub`
    Interpolate smoothed derivatives $\Pi_h(\nabla \boldsymbol{u}_j)$ from nodes `inode` to `igaub`
    Calculate outward unit normal $\boldsymbol{n}$ and test function $\boldsymbol{v}_i$ at `igaub`
    Calculate the product $2\nu \boldsymbol{n} \cdot [\Pi_h(\nabla \boldsymbol{u}_j) + \Pi_h(\nabla \boldsymbol{u}_j)^t] \cdot \boldsymbol{v}_i$ at `igaub` and multiply the result by the weight of the numerical integration
    Assemble result
  end for
end for

Remark 2. In the case of non-overlapping subdomains, for which we only have information on one side of the underlying mesh, the one-layer and multi-layer interpolations are equivalent and are both of first order. The overlapping seems therefore necessary to obtain a higher accuracy.

3.3. Example

We present a simple example of application of the one-layer and multi-layer interpolations to the solution of the Stokes equations. We solve the following system

$$\begin{aligned} &-2\nabla \cdot \boldsymbol{\varepsilon}(\boldsymbol{u}) + 2\boldsymbol{\omega} \times \boldsymbol{u} + \nabla p = \boldsymbol{f}, \\ &\nabla \cdot \boldsymbol{u} = 0, \end{aligned} \tag{12}$$

in a two-dimensional domain Ω made of two concentric circles, where the force $\boldsymbol{f}$ is chosen so that the exact solution of the problem is

$$u_e = 2y[r - 1/2],$$
$$v_e = -2x[r - 1/2],$$
$$p_e = r.$$

with $\boldsymbol{u}_e = [u_e, v_e]^t$ and $r = (x^2 + y^2)^{1/2}$. We construct subdomain 1 of inner diameter 0.5 and outer radius 1, and subdomain 2 of inner radius 0.5 and outer radius 2. In order to test the interpolation technique of the secondary variables, we first solve the problem in subdomain 2 using exact Dirichlet boundary conditions on its boundary, and then update the Neumann condition on the outer circle of subdmomain 1. The solution is a radial Poiseuille-like flow and does not depend on the rotation, although we are going to show that the error of the finite element solution does. The rotation is first chosen to be sufficiently small as we want to avoid any possible instability due to the Coriolis term, so we take $\boldsymbol{\omega} = |\boldsymbol{\omega}|[0, 0, 1]^t$ with $|\boldsymbol{\omega}| = 0.1$. Fig. 5 (top) (left) shows the rate of convergence of the error in subdomain 1 computed for the one-layer and multi-layer interpolations, and confirms that the one-layer interpolation is of first order while the multi-layer interpolation is of second order. Let us analyze what happens if we increase the rotation. Let us denote by $\boldsymbol{u}$ and p an approximate solution, for example a finite element solution or the solution at a certain iteration of a DD method. From the BB condition, we know that there exists $\beta > 0$ such that

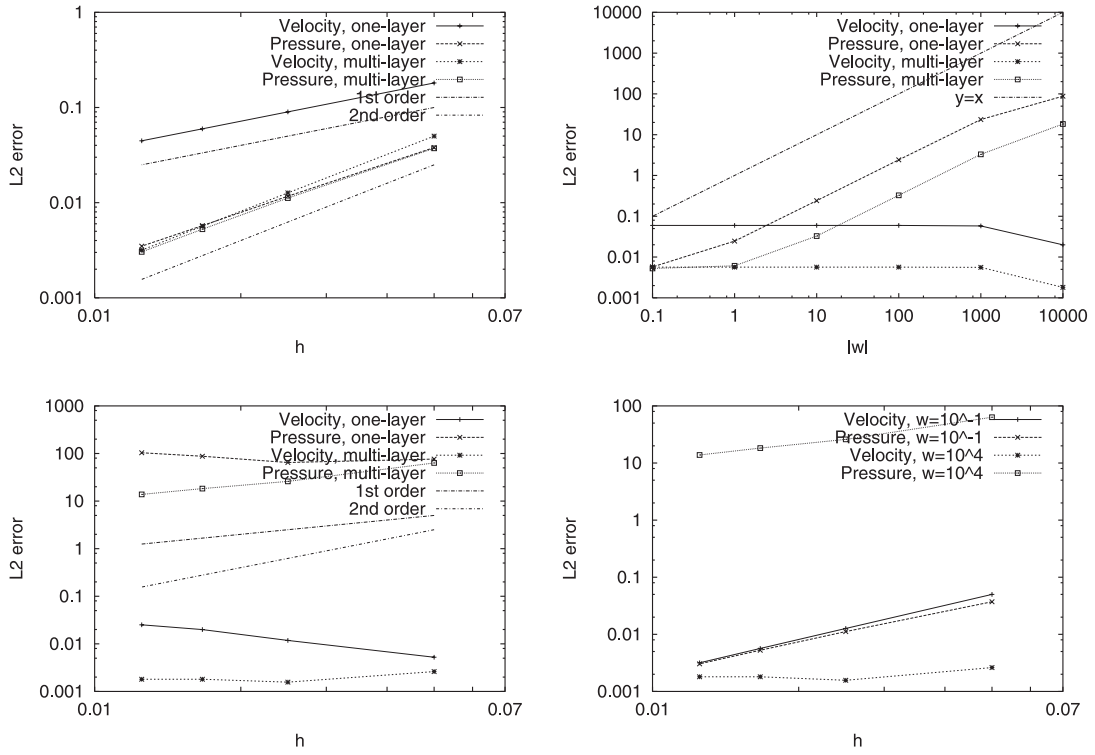

Fig. 5. Concentric circles. L^2 errors. (Top) (Left) $|\boldsymbol{\omega}| = 10^{-1}$. (Top) (Right) $h = 1/60$. (Bot.) (Left) $|\boldsymbol{\omega}| = 10^4$. (Bot.) (Right) Multi-layer interpolation.

$$\|q\|_{P,\Omega} \leqslant \frac{1}{\beta} \sup_{\boldsymbol{v}\in V} \frac{(q, \nabla\cdot\boldsymbol{v})}{\|\boldsymbol{v}\|_{V,\Omega}} \quad \forall q \in Q.$$

Taking $q = p - p_{\mathrm{e}}$, and from Eq. (12) observing that

$$(p - p_{\mathrm{e}}, \nabla\cdot\boldsymbol{v}) = (\nabla(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}), \nabla\boldsymbol{v}) - (2\boldsymbol{\omega}\times(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}), \boldsymbol{v}),$$

we obtain

$$\|p - p_{\mathrm{e}}\|_{P,\Omega} \leqslant \frac{1}{\beta} \sup_{\boldsymbol{v}\in V} \frac{(\nabla(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}), \nabla\boldsymbol{v}) + (2\boldsymbol{\omega}\times(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}), \boldsymbol{v})}{\|\boldsymbol{v}\|_{V,\Omega}} \leqslant \frac{1}{\beta}\left(\sup_{\boldsymbol{v}\in V} \frac{(\nabla(\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}), \nabla\boldsymbol{v})}{\|\boldsymbol{v}\|_{V,\Omega}} + 2|\boldsymbol{\omega}|\|\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}\|_{-1,\Omega}\right).$$

When $\boldsymbol{\omega}$ is high, the second term dominates, i.e.

$$\|p - p_{\mathrm{e}}\|_{P,\Omega} \sim \frac{2|\boldsymbol{\omega}|}{\beta}\|\boldsymbol{u} - \boldsymbol{u}_{\mathrm{e}}\|_{-1,\Omega}, \tag{13}$$

so we expect that the pressure becomes out of control when we have an error in the velocity. Hopefully, when passing Neumann transmission conditions, the error in pressure remains and does not affect the velocity. The mesh convergence for $|\boldsymbol{\omega}| = 10^4$ is shown in Fig. 5 (bot.) (left). We see that the pressure convergence is entirely dominated by the rotation term. In addition, Fig. 5 (top) (right) gives the dependence of the errors with respect to $|\boldsymbol{\omega}|$. The velocity is not negatively affected by the rotation while the error in pressure goes linearly with $|\boldsymbol{\omega}|$, as predicted by Eq. (13). Finally, Fig. 5 (bot.) (right) gives the mesh convergence of the multi-layer interpolation for the pressure and velocity. We observe that the velocity error for $|\boldsymbol{\omega}| = 10^4$ is always below the velocity error for $|\boldsymbol{\omega}| = 10^{-1}$ for the range of mesh sizes studied, while that of the pressure is four orders of magnitude greater.

3.4. Chimera method

In this section we first describe the purpose of the Chimera approach by giving an overview of the possibilities of the method. Then we introduce some terminology and explain the way that the Chimera method can be implemented as an iteration-by-subdomain DD technique. In particular, we build a Chimera method based on Dirichlet/Dirichlet and Dirichlet/Neumann(Robin) couplings on overlapping meshes.

3.4.1. Geometrical coupling and terminology

For the sake of clarity, we assume that the flow we solve only involves one object. The generalization to multi-component flows is straightforward. We first define a background mesh containing all the computational domain, preferably structured. We also generate an independent mesh around the object and dispose it onto the background mesh. This is the patch mesh. The set of the two overset grids is called the *composite grid*, or composite mesh. The idea of the Chimera method is to remove some elements of the background located inside the patch in order to define an *apparent interface*; this task is called *hole cutting*. The hole cutting technique is not described here and the reader is referred for example to [23], or to [31] for the particular method used in this work.

The Chimera method consists in exchanging suitable transmission conditions between the outer boundary of the patch and the apparent interface just defined. The nodes forming the apparent interface are called the *fringe nodes*. In the case of steady subdomains, the only nodes participating to the DD process are the fringe nodes and the interior nodes of the hole, called the *hole nodes*, can be eliminated from the solution process. In the case of moving subdomains, where hole nodes can become fringe nodes at a time step, they cannot be eliminated and the fringe nodes as well as the hole nodes are *interpolation nodes*: they participate to the DD coupling. Finally, some nodes can be excluded from the DD process in order to control the

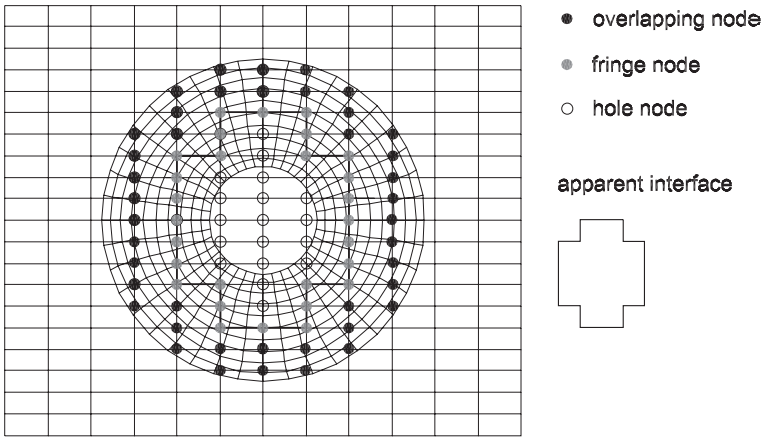

Fig. 6. Chimera method: hole cutting and terminology.

distance between the patch and background interfaces. These nodes are called *overlapping nodes*. This point is treated further on. An example of construction of a Chimera coupling is presented in Fig. 6.

3.4.2. Transmission conditions

In Section 2.3, we studied an iteration-by-subdomain method based on an overlapping Dirichlet/Neumann(Robin) coupling. We now generalize the overlapping Dirichlet/Neumann method applied to the Chimera method, and propose a new Chimera/Neumann coupling (C/N) and Chimera/Robin coupling (C/R). We also propose to study the classical Chimera method, referred to here as Chimera/Dirichlet coupling (C/D). The background mesh is the "Chimera" subdomain for which the velocity is interpolated at the fringe nodes; the patch mesh is assigned a Dirichlet, a Neumann or a Robin transmission condition on its outer boundary. This choice is not arbitrary but practical. We could in fact envisage to assign the apparent interface a Neumann transmission condition. But to do so, we would have to construct explicitly the boundary on the apparent interface and to compute the normal exterior to it. On the contrary, it is relatively easy to prescribe the velocity at the fringe nodes of the background mesh.

"Chimera" is not actually an appropriate term to define an interface type as it generally defines a complete DD method in the scientific literature, but we hope its use in the present context is clear. The C/D and C/N(R) couplings are illustrated in Fig. 7. The required overlap of the C/D is in general greater than that required for mixed transmission conditions, for which, under certain conditions, the overlap is not necessary to achieve convergence. This point is now treated.

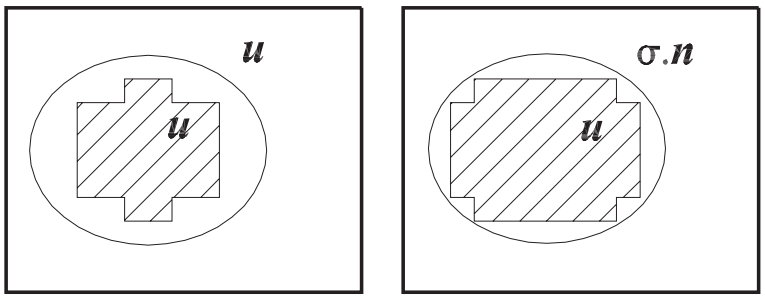

Fig. 7. Chimera method. Hole and variables transmitted. (Left) Chimera/Dirichlet. (Right) Chimera/Neumann(Robin).

3.4.3. The overlap

It is well known that the convergence of iteration-by-subdomain methods based on a Dirichlet/Dirichlet coupling (Schwarz method) depends explicitly on the overlapping length [46]. In [31], we also showed the explicit dependence of the convergence of mixed methods upon the overlapping length of two adjacent subdomains; it is therefore interesting to be able to control the geometric length between interfaces. The algorithm used to ensure a minimum overlap is trivial: if the distance of a background mesh node to the patch interface is lower than the overlapping length desired, then the node cannot participate to the DD coupling, i.e. its degrees of freedom are unknowns of the problem.

Aside from the geometric overlap, a certain number of elements of overlap may be required, which will determine the convergence of the iterative procedure as well as its accuracy. The construction of the C/D coupling requires special care, as a minimum overlap is required to avoid nodes from coinciding. If this becomes the case, the interpolated variable would be frozen at its initial value on the coinciding nodes. In addition, a minimum overlap of one layer of element is needed on each mesh participating to the C/D coupling: this is a sufficient condition to ensure not only continuity of the velocity but also of its derivatives.

This is not the case in the C/N(R) method because the variables interpolated at the interpolation nodes are different from those interpolated at the interface nodes. However, we saw in Section 3.2.3.2 that the overlapping Dirichlet/Neumann(Robin) needs at least one element-layer of overlap to perform the least-square smoothing.

The interpolation nodes that are eliminated to define a given overlap are called overlapping nodes. Fig. 6 shows an example of Chimera coupling where an overlap of one layer of elements on each mesh is achieved. This composite mesh could therefore be used for the C/D method as well as for the C/N and C/R methods using the least-square smoothing to compute the velocity strain rates.

3.4.4. The algorithm

The C/D, C/N and C/R methods fit perfectly into the framework of the Master/Slave coupling described in Section 3.2. When dealing with various unconnected patch grids, the solution on each of these subdomains can be obtained in parallel, while keeping the sequential coupling with the background. To do so, the same color is assigned to the patch subdomains.

4. Numerical examples

In the following examples we consider two types of element, both using equal order interpolation for the velocity and the pressure. The Q1/Q1 element is continuous and bilinear (trilinear in three dimensions) in both velocity and pressure. We also work with the P1/P1 element, continuous and linear in velocity and pressure. These elements do not satisfy the BB condition and therefore they require the use of stabilization described in Section 3.1.

4.1. Vortex shedding behind a cylinder

This example involves the flow past a cylinder, a widely solved benchmark problem. A circular cylinder is immersed in a viscous fluid. The Reynolds number is based on the cylinder diameter D and the prescribed uniform inflow velocity U. The geometry and boundary conditions are shown in Fig. 8. We set $U = 1$ and $D = 1$.

For Re approximately less than 40, two symmetrical eddies develop behind the cylinder. These eddies become unstable at higher Reynolds numbers and periodic vortex shedding occurs, leading to the so-called Von Karman vortex street. We first consider the stationary state at $Re = 30$. As a reference solution, we solve the steady laminar flow on a relatively fine mesh composed of 5400 Q1/Q1 elements, shown in Fig. 9

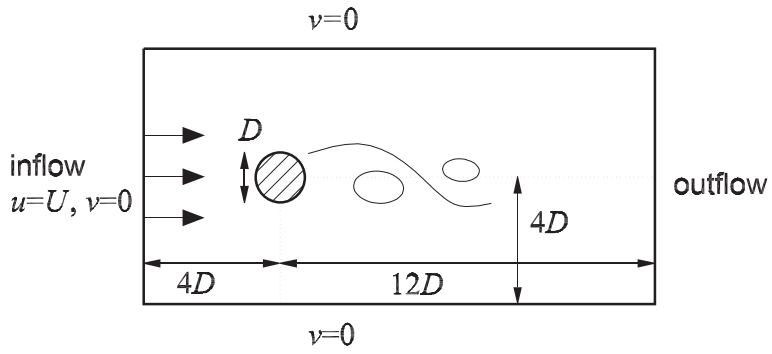

Fig. 8. Vortex shedding. Geometry.

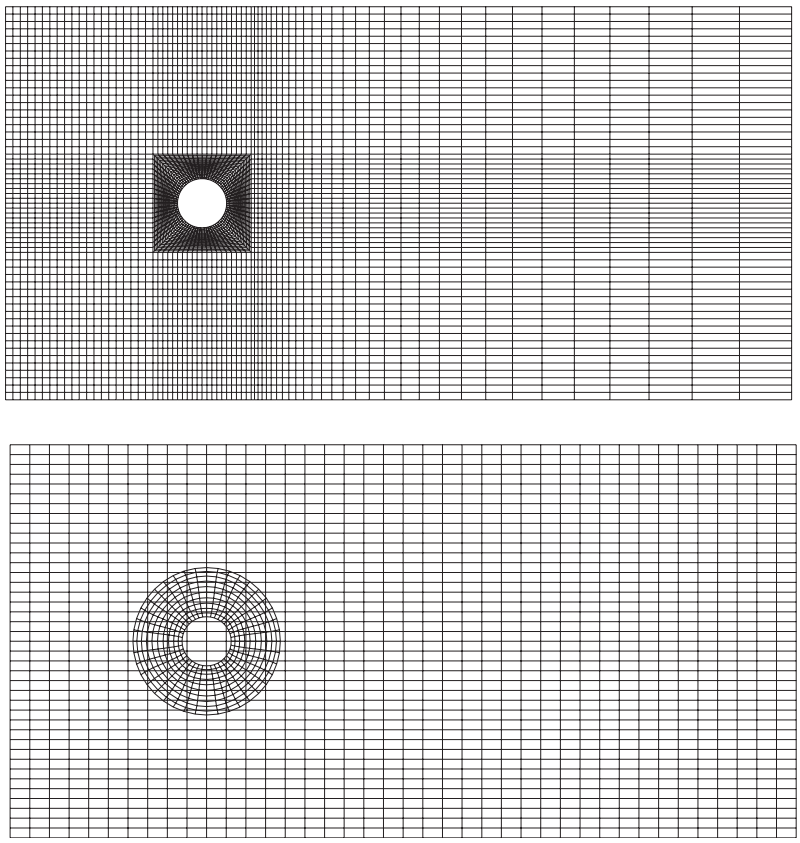

Fig. 9. Vortex shedding. Meshes. (Top) Fine mesh used for one-domain solution. (Bot.) Composite mesh of the Chimera method.

(top). We want to compare here the results obtained with the C/D, C/N and C/R methods. As a background mesh, we use a structured mesh composed of 1600 Q1/Q1 elements. The patch mesh contains the cylinder. Its outer boundary, i.e. the interface of the DD method, is a circle of diameter 3. Its mesh is composed of 400 Q1/Q1 elements. The resulting composite mesh is shown in Fig. 9 (bot.).

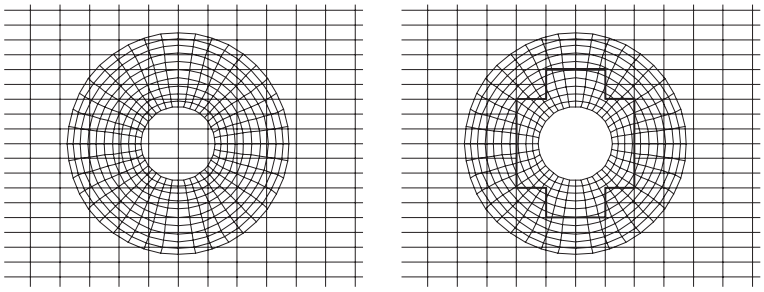

Fig. 10. Vortex shedding. Zoom of meshes. (Left) Composite mesh. (Right) Hole cutting for one element overlap.

Fig. 10 shows a close up of the composite mesh in the cylinder region and the results of the hole cutting operation. The right figure shows the hole created to obtain a one element overlap one each subdomain. This composite mesh will be used for the C/D method as well as for the C/N and C/R method in order to achieve a second-order method.

To solve the stationary problem, we use the Chimera method with one element overlap on each subdomain. Note that when considering the C/D method, the patch subdomain is confined. Therefore, in order to have a well-posed problem on the patch subdomain at each iteration, we apply the interface constraining of the mass conservation [36].

The test we now carry out consists in determining the range of relaxation parameters for which the algorithm converges. To do so, we vary the relaxation parameters of both transmission conditions from 0.1 to 2. The C/D turns out to be the most robust method, i.e. the method for which we have the greatest amplitude in the choice of relaxation parameters to achieve convergence. The C/N does not converge at all, at least for the range of parameters tested. The C/R method converges but for restricted area in the relaxation space, as shown in Fig. 11 (left), where θ_{D} refers to the relaxation parameter of the Dirichlet condition and θ_{R} refers to that of the Robin condition. Fig. 11 (right) compares the convergence histories obtained with the C/D and C/R methods. For the C/D method no relaxation is used while for the C/R method, we use $\theta_{\mathrm{D}} = \theta_{\mathrm{R}} = 0.2$. The figure shows that the convergence of the C/D method looks like monotone while that of the C/R is more unstable. However, the residuals of the Dirichlet data obtained with both methods are of the same order after 30 iterations.

Now, we have previously mentioned in Section 2.4 that a reaction type term in the ADR equation can help mixed DD method to converge. So let us try to solve the transient problem until we obtain a stationary solution, using the backward Euler time integration scheme with a increment step of 0.1. In order to control

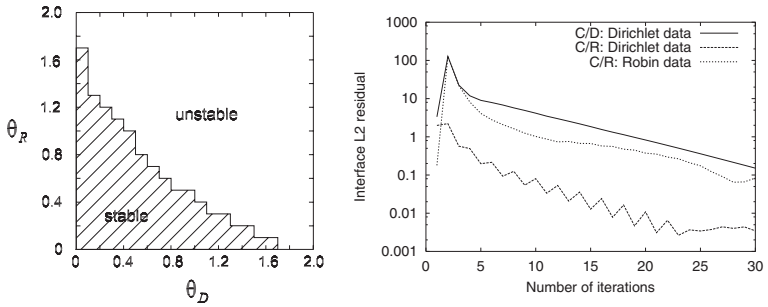

Fig. 11. Vortex shedding. (Left) Stability curve of the C/R method. (Right) Convergence histories of C/D without relaxation and C/R with $\theta_{\mathrm{D}} = \theta_{\mathrm{R}} = 0.2$.

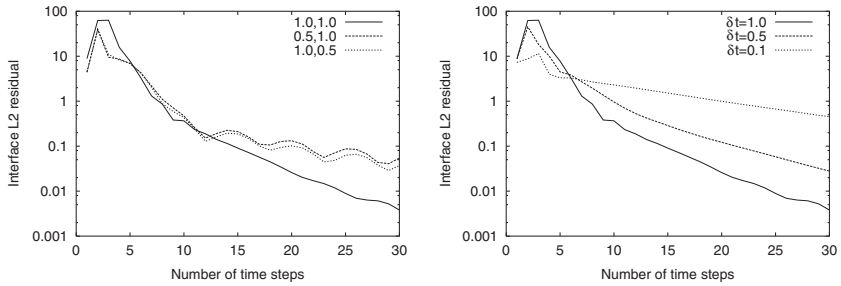

Fig. 12. Vortex shedding. Convergence history of C/N. (Left) For different relaxation parameters (θ_D, θ_N). (Right) For different time steps.

as few parameters as possible in the iterative process, we couple the time, linearization and DD loops. Results are shown in Fig. 12 for the C/N method which in this case converges (to the steady state).

Fig. 13 shows the stationary solutions obtained with the three domain decomposition methods compared with the reference solution obtained on a fine mesh. We observe the good agreement of the mixed methods with the reference solution; on the contrary, the solution of the C/D method differs notably from that of the reference solution. This is attributed to the fact that Dirichlet conditions are much stiffer than Neumann

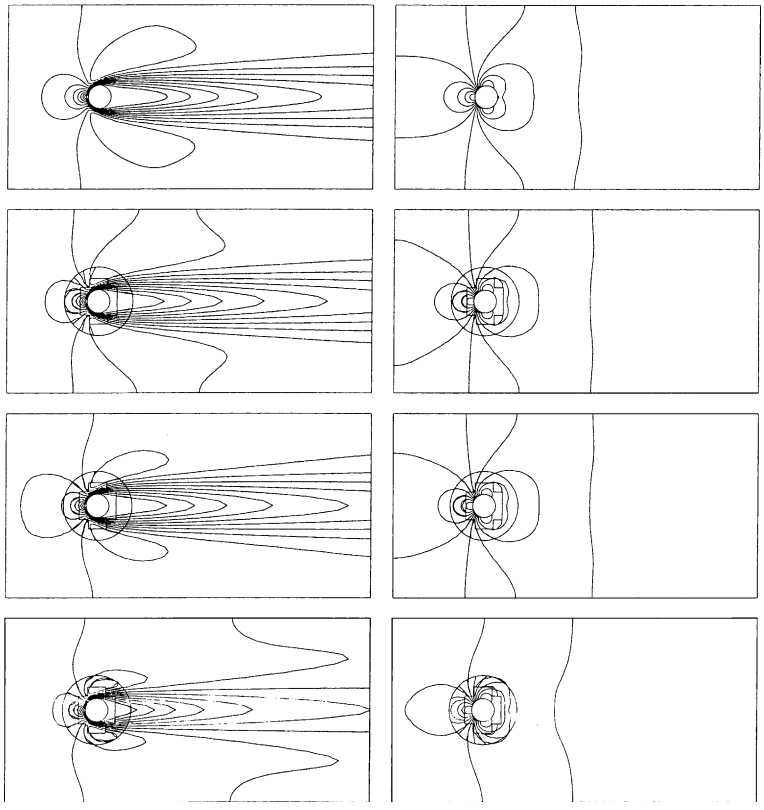

Fig. 13. Vortex shedding. (Left) Velocity module. (Right) Pressure. From top to bottom: one-domain solution, C/N, C/R, C/D.

Table 1
Vortex shedding (amplitude and period of the vertical pressure force)

	One-domain		C/D	C/N	C/R
	Fine mesh	Coarse mesh			
Amplitude	0.170	0.196	0.044	0.110	0.132
Period	5.511	6.622	6.089	6.133	6.089

conditions, i.e. a small error on Dirichlet conditions has much more influence than a small error on the Neumann (or Robin) conditions.

We now go on to the transient case and set $Re = 100$. Although the flow is unstable at this Reynolds number, one can obtain a steady solution. This solution is used as initial condition of the transient simulation, on which we superimpose a small vortex near the cylinder. This is sufficient to trigger the unsteady state. The time integration is carried out with the Crank–Nicholson scheme and $\delta t = 0.1$.

As comparison criteria, we calculate the period and amplitude of the vertical pressure force acting on the cylinder. Numerical references report values of the period between 5.6 and 6.0. See for example [48]. We test the C/D, C/N and C/R methods using a one element overlap. The values of the amplitudes and frequencies are reported in Table 1. As a reference, we also indicate the results obtained with the one-domain simulation using the fine mesh defined earlier and a coarse mesh composed of 1200 Q1/Q1 elements. The C/R method gives the closest results to the one-domain solution, the C/N method being a bit more diffuse. The C/D is much more diffusive than the mixed methods in amplitude and frequency.

4.2. Missile launch from a submarine

In this example we propose to solve the transient and laminar flow around a moving missile [1], using the Chimera/Robin method, i.e. the Chimera method using a Dirichlet/Robin coupling. The geometry is shown in Fig. 14 (left). The missile is moving upward with a constant velocity U. The Reynolds number based on the length H of the missile is

$$Re = \frac{UH}{\nu} = 1000.$$

We set $U = 1$ and $H = 1$. We are going to compare our results to those of Folch [3], obtained with an ALE approach and an explicit flow solver described in [49]. As a background mesh we use a structured mesh of 6000 Q1/Q1 elements, shown in Fig. 15 (top), while the patch mesh is composed of 4173 P1/P1 elements.

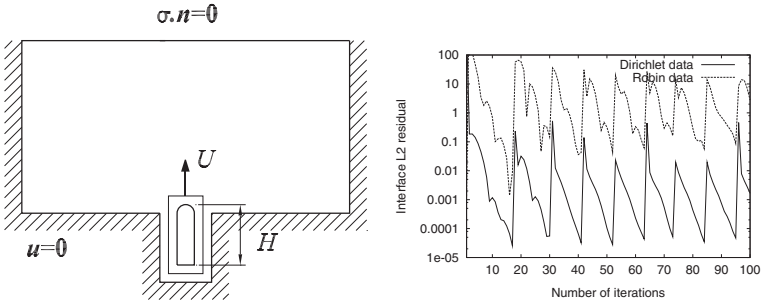

Fig. 14. Missile launch. Geometry and boundary conditions.

According to the C/R method, the velocity of the background mesh is prescribed at the interpolation nodes, while the patch mesh is assigned a Robin transmission condition on its outer boundary. Note that the missile subdomain does not contain any information on its own velocity as the force imposed as transmission condition would be the same in any Galilean frame of reference; all the information on the velocity of the missile is passed through the transmission conditions imposed on the background subdomain, and according to the tensorial transformation described in the Appendix A.

The transient simulation is carried out using the backward Euler scheme with a time step $\delta t = 0.01$. We perform a maximum of 30 domain decomposition/linearization iterations at each time step using as stopping criterion 10^{-4} for the interface residual of the Dirichlet data, the velocity components, given by Eq. (10). The relaxation parameters are $\theta_D = \theta_R = 0.5$. Each problem is solved using a direct solver. The convergence of the problem is shown in Fig. 14 (right).

Fig. 15 shows the composite mesh at time $t = 0.20$, near the missile bottom right corner and at the submarine exit corner.

Fig. 16 presents the velocity vectors obtained at different time steps. They show the development of the vortices created by the suction of air from both sides of the missile.

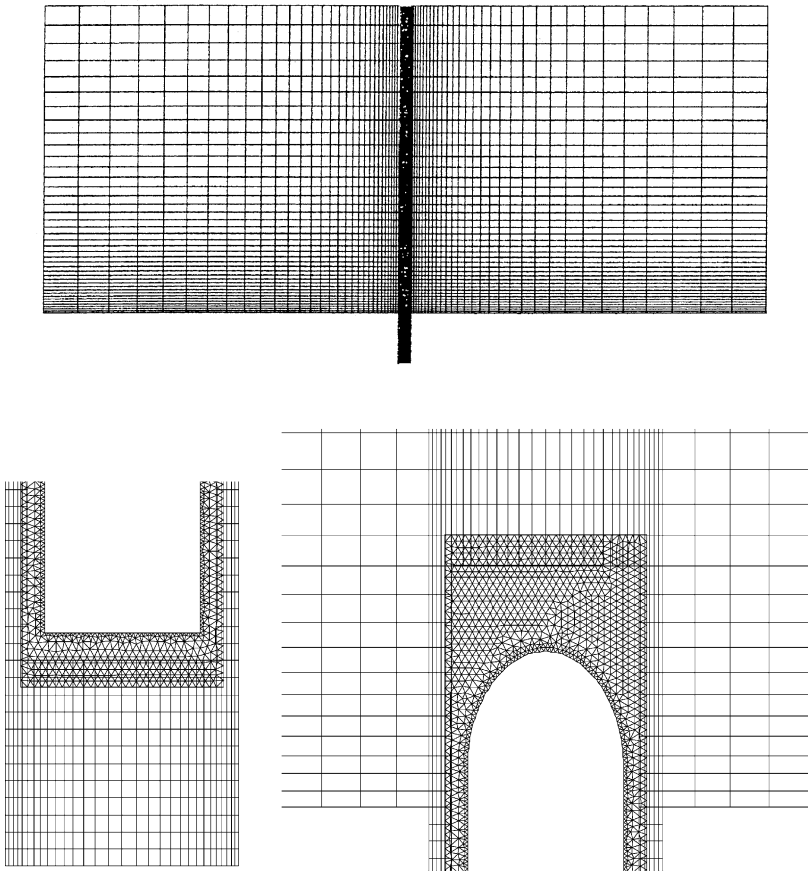

Fig. 15. Missile launch. (Top) Background mesh. (Bot.) Composite mesh at $t = 0.2$. (Bot.) (Left) Bottom right corner of missile. (Bot.) (Right) Submarine exit.

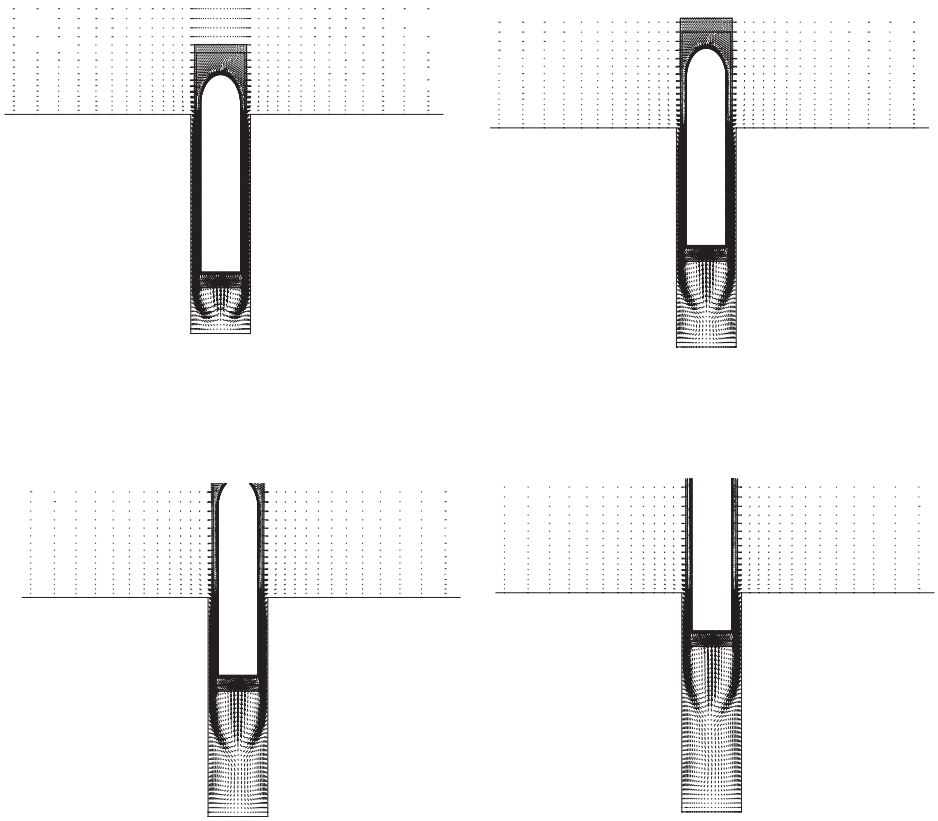

Fig. 16. Missile launch. Velocity vectors. (Top) (Left) $t = 0.2$. (Top) (Right) $t = 0.4$. (Bot.) (Left) $t = 0.6$. (Bot.) (Right) $t = 0.8$.

Figs. 17 and 18 compare the results of the present simulation to those of Folch [3] at different time steps. The first figure presents the streamlines while the next figure presents some pressure contours. They show that both methods give very similar profiles. We notice that an asymmetry develops after the missile leaves the cavity, shown in Figs. 17 (right) and 18 (right). The fact that the asymmetry of the flow develops on the same side for both simulations is just a coincidence. The zig-zags of the streamlines outside of the cavity are due to the large size of the mesh in this region.

4.3. Stirred tank

In this example, we apply the Dirichlet/Neumann method to the solution of a stirred tank. The stirred tank we consider is made of an axial flow impeller and four wall-welded baffles in the tank. The impeller has four pitched blades at a 45° angle designed to draw in the liquid from above and direct it downwards to the bottom of the tank. Actually, the flow is discharged both axially and radially depending on the angle and Reynolds number; for example at low Reynolds numbers the flow is principally radial, as the simulations will show. Stirred tanks are in general very efficient for blending miscible materials and solids suspension. In order to increase the vertical mixing, break up the circular flow around the tank, and possibly to generate turbulence more rapidly, four baffles are disposed around the tank. The baffles are welded to the wall although off-set baffles may be preferable to avoid stagnation zones in the corners.

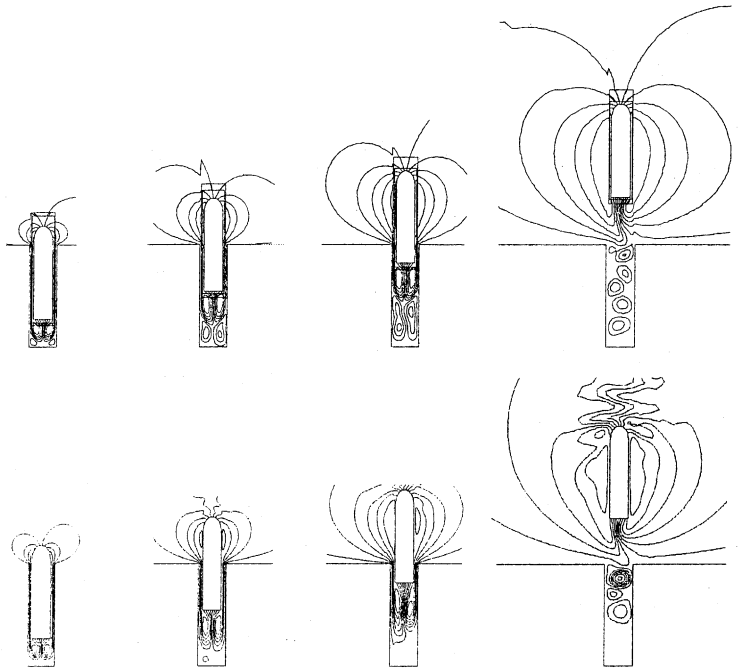

Fig. 17. Missile launch. Streamlines. (Top) Present simulation. (Bot.) Folch's results [3]. From left to right, $t = 0.22$, $t = 0.55$, $t = 0.88$, $t = 1.65$.

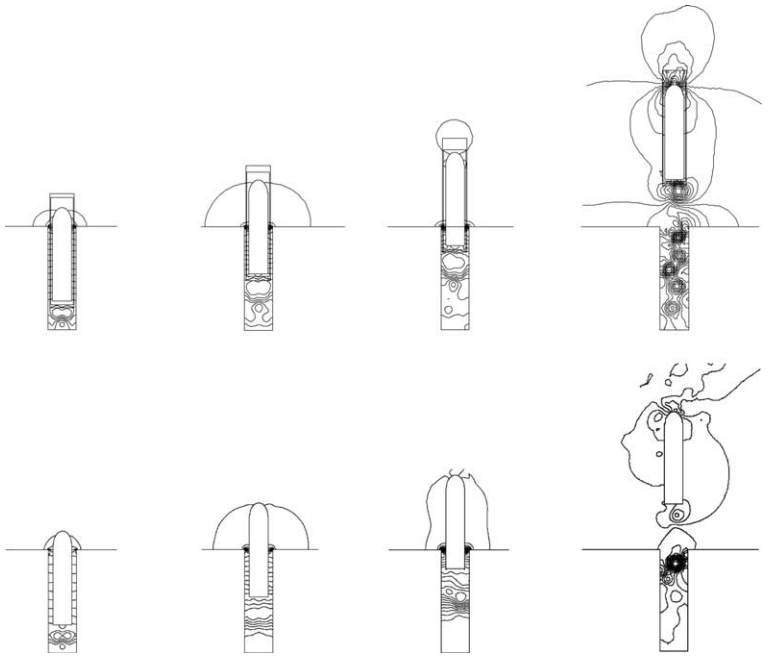

Fig. 18. Missile launch. Pressure. (Top) Present simulation. (Bot.) Folch's results [3]. From left to right, $t = 0.22$, $t = 0.55$, $t = 0.88$, $t = 1.65$.

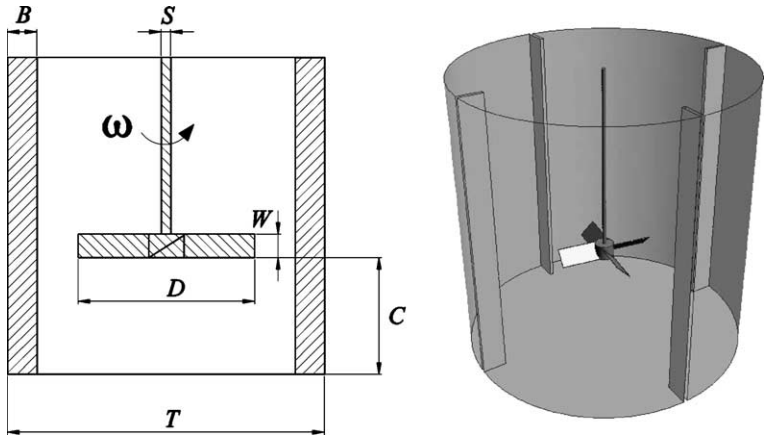

Fig. 19. Stirred tank. Geometry.

The geometry is based on the stirred tank described in [19], and is shown in Fig. 19. The tank has a diameter $T = 0.3$ m, while the impeller diameter is $D = T/3$. The blades have a width $W = D/5$, and the impeller to bottom clearance is $C = T/3$. The four baffles are $B = T/12$ wide.

The non-inertial subdomain is attached to the impeller and is assigned a Neumann transmission condition. The fixed subdomain is the tank and is assigned a Dirichlet transmission condition. The impeller subdomain is meshed with 93332 P1/P1 elements and the tank subdomain with 23135 P1/P1 elements; they are shown in Fig. 20. These meshes are too coarse to describe accurately the physics of the flow, but they are enough to show the performance of the DD strategy we propose. The Dirichlet subdomain is confined so we apply the interface constraining of the mass conservation.

The impeller rotational speed is $N = 225$ r.p.m. which corresponds to an angular velocity $|\omega| = 23.6$ rad/s. The agitator tip speed is $U = |\omega|D/2 = \pi DN = 1.18$ m/s, providing a low agitation. The Reynolds number is defined as:

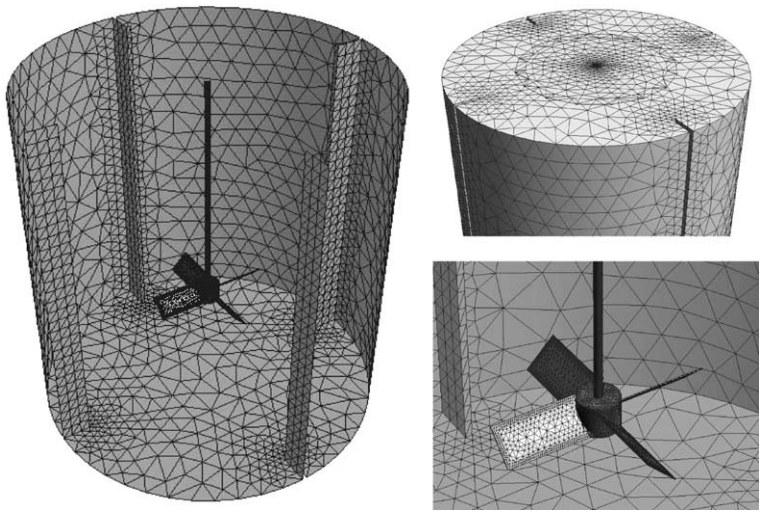

Fig. 20. Stirred tank. Composite mesh. (Left) Overwhole view. (Top) (Right) Top of the tank. (Bot.) (Right) Impeller and baffle.

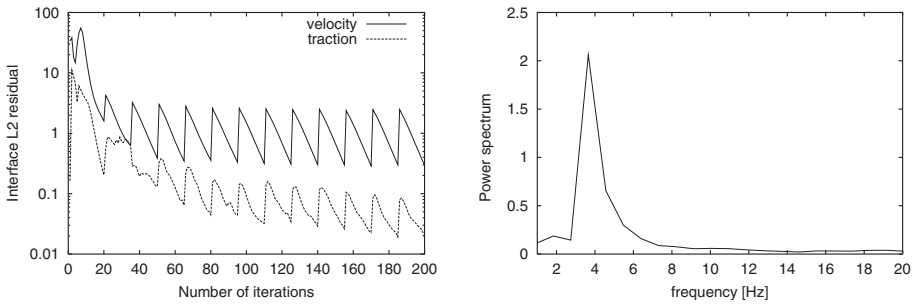

Fig. 21. Stirred tank. (Left) Convergence history. (Right) Power spectrum of the x-viscous force.

$$Re = \frac{ND^2}{v} = 90,$$

while according to this choice the Eckman number is simply $Ek = 1/2Re$. The time integration is carried out using the backward Euler scheme with a time step $\delta t = 0.005$ s. Each problem is solved using an iterative solver (GMRES with diagonal preconditioning).

Fig. 21 (left) shows the convergence of the DD method in the first steps of the simulation. Due to the high viscosity of the fluid, the flow becomes very rapidly periodic. Fig. 21 (right) presents the power spectrum of the x-viscous force exerted on the impeller. We recognize the rotation frequency at 3.75 Hz, but we cannot distinguish any other important frequency.

Fig. 22 shows the pressure contours on the impeller blades. The contours are smooth and confirm the good stabilization of the numerical scheme. On the left blade the pressure is low: this is the suction face which draws the fluid from above. Fig. 23 (top) (left) shows the pressure contours on a vertical cut outlining the low pressure above the impeller and high pressure below the impeller. On the right blade, the pressure is higher and pushed the flow downwards.

The fluid vertical swirl is confirmed by Fig. 23 (top) (right) which shows a vertical cut of the velocity vectors. Fig. 23 (bot.) (left) shows the instantaneous streamlines, winding around the tank from top to bottom and bottom to top. Finally, Fig. 23 (bot.) (right) shows vertical velocity contours on two horizontal cuts.

This example demonstrates the usefulness of coupling domains in relative motion through a domain decomposition strategy. Likewise, it also serves to show the efficiency of the Dirichlet/Neumann coupling when there is is a small overlap between the subdomains involved.

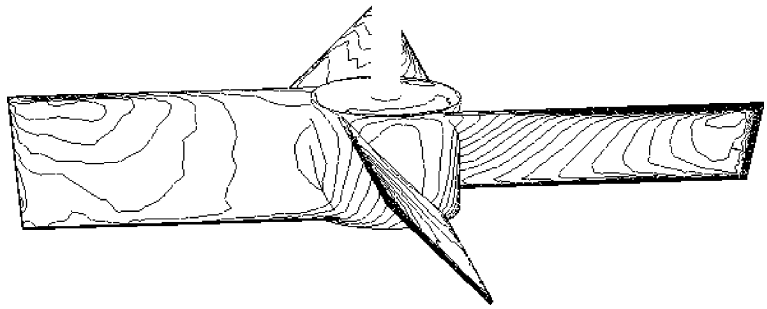

Fig. 22. Stirred tank. Pressure on impeller. Left blade: low pressure. Right blade: high pressure.

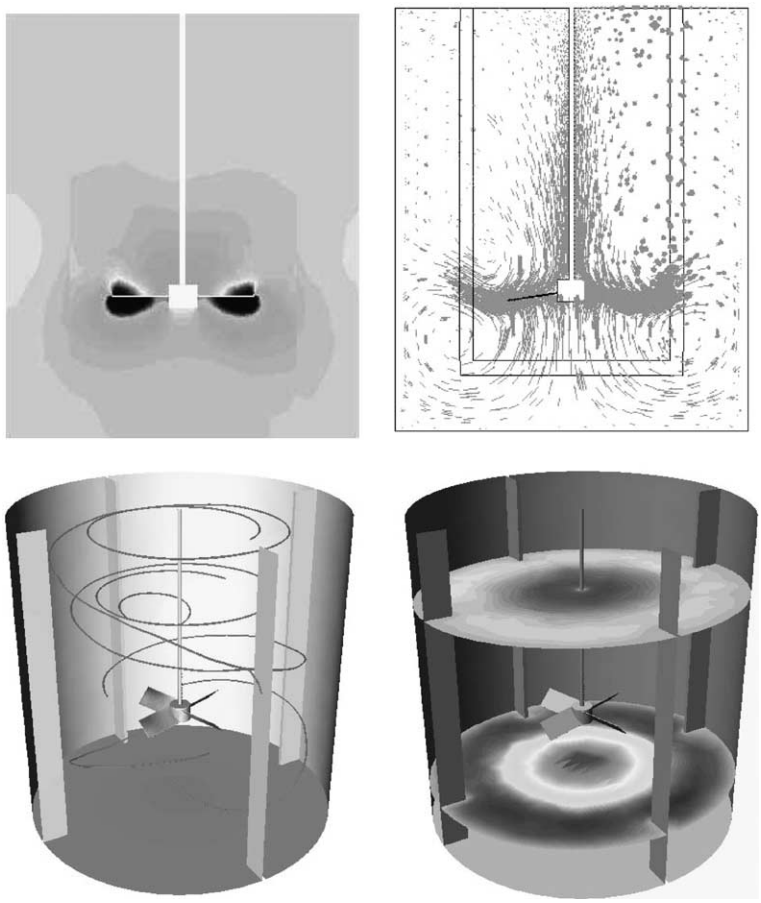

Fig. 23. Stirred tank. (Top) (Left) Pressure. (Top) (Right) Velocity vectors on vertical cut. (Bot.) (Left) Streamlines. (Bot.) (Right) Vertical velocity.

5. Conclusions

We have proposed and studied an iteration-by-subdomain method on overlapping subdomains, based on Dirichlet/Neumann and Dirichlet/Robin transmission conditions, and applied to the solution of the transient Navier–Stokes equations. These methods are extensions of some existing DD methods on disjoint subdomains to the case of overlapping subdomains; the transmission conditions on the interfaces are mixed, i.e. they are of different type on each side of the interfaces; the solutions on the subdomains are coupled iteratively until convergence is achieved.

In the view of a practical implementation for the solution of the Navier–Stokes equations, we built up a Master/Slave algorithm to couple efficiently the numerical solution obtained on different subdomains. A master code is in charge of controlling the iterative process and performing all the necessary operations to leave the slaves unworried. Therefore, very few modifications of the original finite element solver are required. We then discussed the importance of the way the Neumann data is calculated: we identified the need for using the solution from the underlying mesh on both sides of the Neumann-type interface. This is not possible when using disjoint subdomains as the solution is only available on one side. From this remark, we

derived a scheme based on a least-square smoothing of the derivatives. This scheme requires at least a one-element overlap between the subdomains. In order to solve incompressible flows in complex geometries, we introduced a Chimera strategy. We applied the iteration-by-subdomain algorithm to the solution of flows around moving objects by deriving tensorial transformations (see the Appendix A) and an accurate time integration algorithm.

From the numerical examples presented, and also from the numerical experience obtained with other problems not presented here, we may conclude that the Dirichlet/Robin method (and also the associated C/R version) offers the best compromise between accuracy, robustness and flexibility. In particular:

- For flows with convection, it is preferable to the Dirichlet/Neumann method. We saw in the example of Section 4.1 that the C/N did not converge, whereas the C/R method did.
- It is more flexible that the Dirichlet/Dirichlet coupling, since very small (or zero) overlap is possible.
- It is more accurate that the Dirichlet/Dirichlet coupling. In particular, we have observed that it is less diffusive. However, this may in turn affect robustness, since overly diffusive schemes have obviously better numerical behavior in iterative schemes. This was also noted in the example of Section 4.1.

Appendix A. Tensorial transformations

If subdomains i and j are in relative motion, tensorial transformations must be performed each time a variable is obtained in i from j and when a host element is to be found. Let us denote $\boldsymbol{E}_k$ the basis vector in the kth direction of an absolute frame of reference and $\boldsymbol{X}$ the coordinate vector of a point measured in it. Assume we know or we have a way to calculate the translation vector $\boldsymbol{T}_i$ and the rotation matrix $\boldsymbol{\Theta}_i$ of subdomain i as well as those of subdomain j ($\boldsymbol{T}_j$ and $\boldsymbol{\Theta}_j$ respectively), as shown in Fig. 24.

A.1. Expressions for the position, velocity and strain rates

We want to express the position, the velocity and the strain rates in subdomain i in terms of the variables measured in j. We have

$$\boldsymbol{x}_j = \boldsymbol{\Theta}_j(\boldsymbol{X} - \boldsymbol{T}_j),$$
$$\boldsymbol{x}_i = \boldsymbol{\Theta}_i(\boldsymbol{X} - \boldsymbol{T}_i).$$

Knowing that the rotation matrix is orthogonal, we easily get $\boldsymbol{x}_i$ in terms of $\boldsymbol{x}_j$:

$$\boldsymbol{x}_i = \boldsymbol{\Theta}_i(\boldsymbol{\Theta}_j^t \boldsymbol{x}_j + \boldsymbol{T}_j - \boldsymbol{T}_i). \tag{A.1}$$

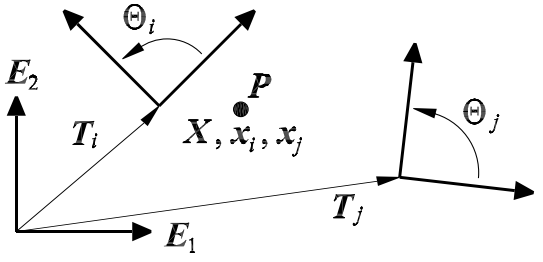

Fig. 24. Two moving frames of reference in the absolute one.

Differentiating this equation with respect to time, we find the velocity $\boldsymbol{u}_i$ in terms of the velocity $\boldsymbol{u}_j$ measured in j:

$$\boldsymbol{u}_i = \dot{\boldsymbol{x}}_i = \dot{\boldsymbol{\Theta}}_i(\boldsymbol{\Theta}_j^t \boldsymbol{x}_j + \boldsymbol{T}_j - \boldsymbol{T}_i) + \boldsymbol{\Theta}_i(\dot{\boldsymbol{\Theta}}_j^t \boldsymbol{x}_j + \boldsymbol{\Theta}_j^t \boldsymbol{u}_j + \dot{\boldsymbol{T}}_j - \dot{\boldsymbol{T}}_i), \tag{A.2}$$

where $\dot{(\cdot)} = \mathrm{d}(\cdot)/\mathrm{d}t$ and $\dot{\boldsymbol{T}}_j$ and $\dot{\boldsymbol{T}}_i$ are the velocities of subdomains j and i measured in the absolute frame of reference $\boldsymbol{E}_k$.

We now derive the transformation of the strain rates, i.e. we want to express the strain rates $\boldsymbol{\varepsilon}_i(\boldsymbol{u}_i)$ measured in subdomain i as a function of the strain rates $\boldsymbol{\varepsilon}_j(\boldsymbol{u}_j)$ measured in subdomain j. We have

$$\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_i} = \frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_j}\frac{\partial \boldsymbol{x}_j}{\partial \boldsymbol{x}_i}.$$

By substituting Eq. (A.2), and knowing that

$$\frac{\partial \boldsymbol{x}_j}{\partial \boldsymbol{x}_i} = \boldsymbol{\Theta}_j \boldsymbol{\Theta}_i^t,$$

we obtain

$$\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_i} = \dot{\boldsymbol{\Theta}}_i \boldsymbol{\Theta}_j^t \boldsymbol{\Theta}_j \boldsymbol{\Theta}_i^t + \boldsymbol{\Theta}_i(\dot{\boldsymbol{\Theta}}_j^t \boldsymbol{\Theta}_j)\boldsymbol{\Theta}_i^t + \boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t \frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{x}_j}\boldsymbol{\Theta}_j \boldsymbol{\Theta}_i^t.$$

Due to the orthogonality of the rotation matrices, $\boldsymbol{\Theta}_j^t \boldsymbol{\Theta}_j = I$ so the last equation gives

$$\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_i} = \dot{\boldsymbol{\Theta}}_i \boldsymbol{\Theta}_i^t + \boldsymbol{\Theta}_i(\dot{\boldsymbol{\Theta}}_j^t \boldsymbol{\Theta}_j)\boldsymbol{\Theta}_i^t + \boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t \frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{x}_j}\boldsymbol{\Theta}_j \boldsymbol{\Theta}_i^t,$$

and

$$\left(\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_i}\right)^t = \boldsymbol{\Theta}_i \dot{\boldsymbol{\Theta}}_i^t + \boldsymbol{\Theta}_i(\boldsymbol{\Theta}_j^t \dot{\boldsymbol{\Theta}}_j)\boldsymbol{\Theta}_i^t + \boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t \left(\frac{\partial \boldsymbol{u}_j}{\partial \boldsymbol{x}_j}\right)^t \boldsymbol{\Theta}_j \boldsymbol{\Theta}_i^t.$$

Now we add up the latter two equations and divide the result by two to obtain the equation for the strain rate:

$$\begin{aligned}
\boldsymbol{\varepsilon}_i(\boldsymbol{u}_i) &= \frac{1}{2}\left[\left(\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_i}\right) + \left(\frac{\partial \boldsymbol{u}_i}{\partial \boldsymbol{x}_i}\right)^t\right] \\
&= \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{\Theta}_i \boldsymbol{\Theta}_i^t) + \boldsymbol{\Theta}_i \frac{\mathrm{d}}{\mathrm{d}t}(\boldsymbol{\Theta}_j^t \boldsymbol{\Theta}_j)\boldsymbol{\Theta}_i^t + (\boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t)\boldsymbol{\varepsilon}_j(\boldsymbol{u}_j)(\boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t)^t \\
&= \frac{\mathrm{d}I}{\mathrm{d}t} + \boldsymbol{\Theta}_i \frac{\mathrm{d}I}{\mathrm{d}t}\boldsymbol{\Theta}_i^t + (\boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t)\boldsymbol{\varepsilon}(\boldsymbol{u}_j)(\boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t)^t.
\end{aligned}$$

The first two terms are zero so we finally find that the velocity strain rate tensor transforms like:

$$\boldsymbol{\varepsilon}_i(\boldsymbol{u}_i) = (\boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t)\boldsymbol{\varepsilon}_j(\boldsymbol{u}_j)(\boldsymbol{\Theta}_i \boldsymbol{\Theta}_j^t)^t. \tag{A.3}$$

We observe that the velocity strain rate undergoes a rotation but no scaling, contrary to the velocity. We can also check that this expression is symmetric.

A.2. A second-order time integration of the basis vectors

In order to close the transformation of the position, velocity and strain rates expressed by Eqs. (A.1), (A.2) and (A.3), we need to compute the rotation matrices $\boldsymbol{\Theta}_j$ and $\boldsymbol{\Theta}_i$ at each time. Let us denote by $\boldsymbol{\Theta}$ the

rotation matrix of a frame of reference of basis vectors $\boldsymbol{e}_k$s with respect to the absolute frame of reference, given by

$$\boldsymbol{\Theta} = [\, \boldsymbol{e}_1 \quad \boldsymbol{e}_2 \quad \boldsymbol{e}_3 \,]^t$$

and assume we know the angular velocity vector $\boldsymbol{\omega}$ of the frame, which is a function of time. At each instant, the change of the basis vectors satisfies:

$$\dot{\boldsymbol{e}}_k(t) = \boldsymbol{\omega}(t) \times \boldsymbol{e}_k(t) = \boldsymbol{W}(t)\boldsymbol{e}_k(t), \tag{A.4}$$

where $W_{pq} = -\varepsilon_{pqr}\omega_r$ is given in the basis $\boldsymbol{E}_k$; ε_{pqr} is the permutation (alternating) tensor with value zero if two indices are repeated, and with value 1 or -1 if p, q, r are in cyclic order or not, respectively. In the following, we use the matrix form given by Eq. (A.4). Let us consider a partition $0 = t^0 < t^1 < \cdots < t^N = T$ of the time interval $[0, T]$ of interest. In order to integrate Eq. (A.4), we propose the following approximation:

$$\dot{\tilde{\boldsymbol{e}}}_k(t) = \left(\boldsymbol{W}^n + \tfrac{1}{2}\dot{\boldsymbol{W}}^n\,\delta t\right)\tilde{\boldsymbol{e}}_k(t), \quad \text{for } t^n \leqslant t \leqslant t^{n+1}, \tag{A.5}$$

where superscript n denotes variables considered at time t^n, the tilde indicates that the solution is approximated and $\delta t = t^{n+1} - t^n$. We are now going to show that the approximation given by Eq. (A.5) is of second order in time. By direct integration of Eq. (A.5), we find that

$$\tilde{\boldsymbol{e}}_k(t^{n+1}) = \exp\left(\boldsymbol{W}^n\,\delta t + \tfrac{1}{2}\dot{\boldsymbol{W}}^n\,\delta t^2\right)\tilde{\boldsymbol{e}}_k(t^n). \tag{A.6}$$

Let us develop the exact solution of Eq. (A.4) in Taylor series around time t^n:

$$\begin{aligned}\boldsymbol{e}_k(t^{n+1}) &= \boldsymbol{e}_k(t^n) + \dot{\boldsymbol{e}}_k(t^n)\delta t + \tfrac{1}{2}\ddot{\boldsymbol{e}}_k(t^n)\delta t^2 + \mathcal{O}(\delta t^3)\\ &= [\boldsymbol{I} + \boldsymbol{W}^n\,\delta t + \tfrac{1}{2}\dot{\boldsymbol{W}}^n\,\delta t^2 + \tfrac{1}{2}(\boldsymbol{W}^n)^2\,\delta t^2]\boldsymbol{e}_k(t^n) + \mathcal{O}(\delta t^3).\end{aligned}$$

Performing the same expansion for the approximate solution given by Eq. (A.6), we get

$$\tilde{\boldsymbol{e}}_k(t^{n+1}) = [\boldsymbol{I} + \boldsymbol{W}^n\,\delta t + \tfrac{1}{2}\dot{\boldsymbol{W}}^n\,\delta t^2 + \tfrac{1}{2}(\boldsymbol{W}^n)^2\,\delta t^2 + \mathcal{O}(\delta t^3)]\tilde{\boldsymbol{e}}_k(t^n) + \mathcal{O}(\delta t^3).$$

We simplify the latter two equations by introducing

$$\boldsymbol{A}^n = \boldsymbol{I} + \boldsymbol{W}^n\,\delta t + \tfrac{1}{2}\dot{\boldsymbol{W}}^n\,\delta t^2 + \tfrac{1}{2}(\boldsymbol{W}^n)^2\,\delta t^2,$$

so that we have

$$\begin{aligned}\boldsymbol{e}_k(t^{n+1}) &= \boldsymbol{A}^n\boldsymbol{e}_k(t^n) + \mathcal{O}(\delta t^3),\\ \tilde{\boldsymbol{e}}_k(t^{n+1}) &= \boldsymbol{A}^n\tilde{\boldsymbol{e}}_k(t^n) + \mathcal{O}(\delta t^3).\end{aligned}$$

Therefore

$$\begin{aligned}\tilde{\boldsymbol{e}}_k(t^{n+1}) - \boldsymbol{e}_k(t^{n+1}) &= \boldsymbol{A}^n(\tilde{\boldsymbol{e}}_k(t^n) - \boldsymbol{e}_k(t^n)) + \mathcal{O}(\delta t^3)\\ &= \boldsymbol{A}^n\boldsymbol{A}^{n-1}(\tilde{\boldsymbol{e}}_k(t^{n-1}) - \boldsymbol{e}_k(t^{n-1})) + \mathcal{O}(\delta t^3) + \mathcal{O}(\delta t^3)\\ &\;\;\vdots\\ &= \boldsymbol{A}^n\boldsymbol{A}^{n-1}\cdots\boldsymbol{A}^0(\tilde{\boldsymbol{e}}_k(t^0) - \boldsymbol{e}_k(t^0)) + \mathcal{O}(\delta t^2).\end{aligned}$$

Assuming the basis vectors are given at $t = 0$, we have $\tilde{\boldsymbol{e}}_k(t^0) - \boldsymbol{e}_k(t^0) = 0$. Therefore, we have that $\tilde{\boldsymbol{e}}_k(t^{n+1}) - \boldsymbol{e}_k(t^{n+1}) = \mathcal{O}(\delta t^2)$.

In order to find the $\tilde{\boldsymbol{e}}_k$s at t^{n+1}, we apply Eq. (A.6) recursively:

$$\tilde{\boldsymbol{e}}_k(t^{n+1}) = \exp\left[\sum_{m=0}^{n}\left(\boldsymbol{W}^m\,\delta t + \frac{1}{2}\dot{\boldsymbol{W}}^m\delta t^2\right)\right]\boldsymbol{E}_k \tag{A.7}$$

$$= (\boldsymbol{\Theta}^n)^t\boldsymbol{E}_k \quad \text{(by definition)}, \tag{A.8}$$

where we have assumed that $\tilde{\boldsymbol{e}}_k^0 = \boldsymbol{E}_k$. The last expression for the rotation matrix is not convenient, so we try to derive a nicer equation for the coefficients of $\boldsymbol{\Theta}^n$ at time t^n. By definition, we have

$$W_{pq}^m = -\varepsilon_{pqr}\omega_r^m,$$
$$\dot{W}_{pq}^m = -\varepsilon_{pqr}\dot{\omega}_r^m.$$

Let $\boldsymbol{B}$ be the argument matrix of the exponential function of Eq. (A.7), that is,

$$\boldsymbol{B} = \sum_{m=0}^{n}\left(\boldsymbol{W}^m\,\delta t + \frac{1}{2}\dot{\boldsymbol{W}}^m\,\delta t^2\right),$$

whose coefficients are

$$B_{pq} = -\varepsilon_{pqr}\sum_{m=0}^{n}\left(\omega_r^m\,\delta t + \frac{1}{2}\dot{\omega}_r^m\,\delta t^2\right).$$

We define the vector $\boldsymbol{r}^n$ and the unit vector $\hat{\boldsymbol{r}}^n$ as

$$\boldsymbol{r}^n = \sum_{m=0}^{n}\left(\boldsymbol{\omega}^m\,\delta t + \frac{1}{2}\dot{\boldsymbol{\omega}}^m\,\delta t^2\right),$$
$$\hat{\boldsymbol{r}}^n = \frac{\boldsymbol{r}^n}{|\boldsymbol{r}^n|},$$

so the coefficients of matrix $\boldsymbol{B}$ become

$$B_{pq} = -\varepsilon_{pqr}\hat{\boldsymbol{r}}_r^n|\boldsymbol{r}^n|.$$

Let us introduce a matrix $\boldsymbol{C}$ and a scalar θ defined by

$$C_{pq} = -\varepsilon_{pqr}\hat{\boldsymbol{r}}_r^n,$$
$$\theta = |\boldsymbol{r}^n|. \tag{A.9}$$

According to these definitions, Eq. (A.7) can be re-written as

$$\tilde{\boldsymbol{e}}_k(t^{n+1}) = \exp(\theta\boldsymbol{C})\boldsymbol{E}_k.$$

In addition, it can be shown that for a matrix $\boldsymbol{C}$ given by (A.9) with $\hat{\boldsymbol{r}}^n$ being a unit vector, we have

$$(\exp(\theta\boldsymbol{C}))_{pq} = \hat{\boldsymbol{r}}_p^n\hat{\boldsymbol{r}}_q^n + (\delta_{pq} - \hat{\boldsymbol{r}}_p^n\hat{\boldsymbol{r}}_q^n)\cos\theta + C_{pq}\sin\theta$$
$$= \hat{\boldsymbol{r}}_p^n\hat{\boldsymbol{r}}_q^n + (\delta_{pq} - \hat{\boldsymbol{r}}_p^n\hat{\boldsymbol{r}}_q^n)\cos|\boldsymbol{r}^n| - \varepsilon_{pqr}\hat{\boldsymbol{r}}_r^n\sin|\boldsymbol{r}^n|.$$

By definition of the rotation matrix $\boldsymbol{\Theta}^n$ we have

$$(\boldsymbol{\Theta}^n) = (\exp(\theta\boldsymbol{C}))^t,$$

so the coefficients of the previous exponential form are given by:

$$\Theta_{pq}^n = \hat{\boldsymbol{r}}_p^n\hat{\boldsymbol{r}}_q^n + (\delta_{pq} - \hat{\boldsymbol{r}}_p^n\hat{\boldsymbol{r}}_q^n)\cos|\boldsymbol{r}^n| + \varepsilon_{pqr}\hat{\boldsymbol{r}}_r^n\sin|\boldsymbol{r}^n|.$$

We recognize here the expression for the matrix coefficient of a rotation through an angle $|\boldsymbol{r}^n|$ about an axis whose direction is given by the unit vector $\hat{\boldsymbol{r}}$; see for example [47].

The derivative of the rotation function is given by

$$\dot{\boldsymbol{\Theta}}^n = \left[\dot{\tilde{\boldsymbol{e}}}_1(t^n) \quad \dot{\tilde{\boldsymbol{e}}}_2(t^n) \quad \dot{\tilde{\boldsymbol{e}}}_3(t^n)\right]^t.$$

Using (A.5) evaluated at time t^{n+1} and (A.8) it is found that

$$\dot{\tilde{\boldsymbol{e}}}_k(t^{n+1}) = \left(\boldsymbol{W}^n + \tfrac{1}{2}\dot{\boldsymbol{W}}^n \delta t\right)(\boldsymbol{\Theta}^n)^t \boldsymbol{E}_k,$$

from where it follows that

$$\dot{\boldsymbol{\Theta}}^{n+1} = \boldsymbol{\Theta}^n \left(\boldsymbol{W}^n + \tfrac{1}{2}\dot{\boldsymbol{W}}^n \delta t\right)^t.$$

References

[1] A. Masud, T.J.R. Hughes, A space–time Galerkin/least-squares finite element formulation of the Navier–Stokes equations for moving domain problems, Comput. Methods Appl. Mech. Engrg. 146 (1997) 91–126.

[2] E. Mestreau, R. Lohner, S. Aita, TGV tunnel entry simulations using a finite element code with automatic remeshing, AIAA Paper 93-0890, 1993.

[3] A. Folch, A numerical formulation to solve the ALE Navier–Stokes equations applied to the withdrawal of magma chambers, Ph.D. thesis, Universitat Politécnica de Catalunya, Barcelona (Spain), 2000.

[4] M. Zhu, T. Fusegi, A. Tabbal, E. Mestreau, H. Malanda, D. Vinteler, A. Goto, M. Nohmi, A tetrahedral finite-element method for 3d numerical modeling and entire simulation of unsteady turbulent flow in a mixed-flow pump, in: ASME Fluids Engineering Division Summer Meeting, number FEDSM98-4879, Washington, June 1998.

[5] A. Huerta, W.K. Liu, Viscous flow with large free surface motion, Comput. Methods Appl. Mech. Engrg. 69 (1988) 277–324.

[6] A. Soulaimani, M. Fortin, D. Dhatt, Y. Ouellet, Finite element simulation of two and three dimensional free surface flows, Comput. Methods Appl. Mech. Engrg. 89 (1991) 265–296.

[7] J. Donéa, P. Fasoli-Stella, S. Giuliani, Lagrangian and Eulerian finite element techniques for transient fluid structure interaction problems, in: Transactions of the Fourth International Conference of Structural Mechanics in Reactor Technology, number paper Bl/2, 1977.

[8] R. Glowinski, T.-W. Pan, J. Périaux, Fictitious domain methods for the Dirichlet problem and its generalization to some flow problems, in: K. Morgan, E. Oñate, J. Périaux, J. Péraire, O.C. Zienkiewicz (Eds.), Finite Element in Fluids, New Trends and Applications, Pineridge Press, Barcelona, Spain, 1993, pp. 347–368.

[9] R. Glowinski, T.-W. Pan, J. Périaux, A fictitious domain method for Dirichlet problems and applications, Comput. Methods Appl. Mech. Engrg. 111 (1994) 203–303.

[10] F. Bertrand, P.A. Tanguy, F. Thibault, A three-dimensional fictitious domain method for incompressible fluid flow problems, Int. J. Numer. Meth. Fluids 25 (1997) 719–736.

[11] R. Glowinski, T.-W. Pan, J. Périaux, On a domain embedding method for flow around moving rigid bodies, in: P.E. Bjørstad, M. Espedal, D. Keyes (Eds.), Ninth International Conference of Domain Decomposition Methods, ddm.org, 1998, Proceedings from the Nineth International Conference, June 1996, Bergen, Norway.

[12] T.-W. Pan, Numerical simulation of the motion of a ball falling in an incompressible viscous fluid, C.R. Acad. Sci. Paris 327 (Série IIb) (1999) 1035–1038.

[13] R. Glowinski, T.-W. Pan, T.I. Hesla, D.D. Joseph, J. Périaux, A distributed Lagrange multiplier/fictitious domain method for flows around moving rigid bodies: Application to particulate flow, Int. J. Numer. Meth. Fluids 30 (1999) 1043–1066.

[14] L.H. Juárez, Numerical simulation of the sedimentation of an elliptic body in an incompressible viscous fluid, C.R. Acad. Sci. Paris 329 (Série IIb) (2001) 221–224.

[15] R. Glowinski, T.-W. Pan, J. Kearsley, J. Périaux, Numerical simulation and optimal shape for viscous flow by a fictitious domain method, Int. J. Numer. Meth. Fluids 20 (1995) 685–711.

[16] C. Bernardi, Y. Maday, A.T. Patera, A new nonconforming approach to domain decomposition: the mortar element method, in: H. Brezis, J.-L. Lions (Eds.), Nonlinear Partial Differential Equations and Their Applications, Collège de France Seminar, vol. XI, 1994, pp. 13–51.

[17] A. Ben Abdallah, J.-L. Guermond, A fully parallel mortar finite element projection method for the solution of the unsteady Navier–Stokes equations, in: Proceedings of the Third ECCOMAS Computational Fluid Dynamics Conference, Paris (France), John Wiley & Sons Ltd., 1996, pp. 852–858.

[18] G. Anagnostou, Y. Maday, C. Mavriplis, A.T. Patera, On the mortar element method: Generalizations and implementation, in: T.F. Chan, R. Glowinski, J. Périaux, O.B. Widlund, Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, Houston, USA, March 1989, SIAM, Philadelphia, 1990, pp. 157–173.
[19] A. Bakker, R.D. Laroche, M.H. Wang, R.V. Calabrese, Sliding mesh simulation of laminar flow in stirred reactors, Trans. IChemE 74 (Part A) (1997) 42–44.
[20] F. Longatte, J.-L. Kuen, Analysis of rotor–stator-circuit interactions in a centrifugal pump, in: C.P. Kleijn, V.V. Kudriavtsev, W. Cheng, S. Kawano (Eds.), Proceedings of the Third ASME/JSME Joint Fluids Engineering Conference, number FEDSM99-6866 in Computational Technologies for Fluid/Thermal/Chemical Systems with Industrial Applications, San Francisco, July 1999.
[21] M. Behr, T. Tezduyar, The shear-slip mesh update method, Comput. Methods Appl. Mech. Engrg. 174 (1999) 261–274.
[22] R.L. Meakin, N.E. Suhs, Unsteady aerodynamic simulation of multiple bodies in relative motion, 1989, AIAA Paper 89-1996-CP.
[23] Z.J. Wang, V. Parthasarathy, A fully automated Chimera methodology for multiple moving body problems, Int. J. Numer. Meth. Fluids 33 (2000) 919–938.
[24] N.C. Prewitt, D.M. Belk, W. Shyy, Parallel computing of overset grids for aerodynamic problems with moving objects, J. Progr.: Aero. Sci. 36 (2000) 117–172.
[25] G.K. Batchelor, An Introduction to Fluid Dynamics, Cambridge University Press, 1970.
[26] R. Codina, O. Soto, Finite element implementation of two-equation and algebraic stress turbulence models for steady incompressible flows, Int. J. Numer. Meth. Fluids 30 (1999) 309–333.
[27] J.-L. Lions, R. Dautrey, Mathematical Analysis and Numerical Methods for Science and Technology, in: Functional and Variational Methods, vol. 2, Springer-Verlag, 2000.
[28] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, Springer-Verlag, 1994.
[29] A. Quarteroni, A. Valli, Domain Decomposition Methods for Partial Differential Equations, Oxford Science, 1999.
[30] G. Houzeaux, R. Codina, An iteration-by-subdomain overlapping Dirichlet/Robin domain decomposition method for advection–diffusion problems, J. Comput. Appl. Math., in press.
[31] G. Houzeaux, A Geometrical Domain Decomposition Method in Computational Fluid Dynamics. Ph.D. thesis, Universitat Politécnica de Catalunya, Barcelona (Spain), 2001.
[32] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Springer-Verlag, 1991.
[33] T.J.R. Hughes, Multiscale phenomena: Green's functions, the Dirichlet-to-neumann formulation, subgrid scale models, bubbles and the origins of stabilized methods, Comput. Methods Appl. Mech. Engrg. 127 (1995) 387–401.
[34] R. Codina, A stabilized finite element method for generalized stationary incompressible flows, Comput. Methods Appl. Mech. Engrg. 190 (2001) 2681–2706.
[35] R. Codina, On stabilized finite element methods for linear systems of convection–diffusion–reaction equations, Comput. Methods Appl. Mech. Engrg. 188 (2000) 61–82.
[36] G. Houzeaux, R. Codina, Transmission conditions with constraints in finite element domain decomposition method for flow problems, Commun. Numer. Meth. Engrg. 17 (2001) 179–190.
[37] D.N. Arnold, F. Brezzi, B. Cokckburn, D. Marini, Discontinuous Galerkin methods for elliptic problems, in: Discontinuous Galerkin Methods, Lecture Notes in Computational Science and Engineering, vol. 11, Springer, 2000, pp. 89–200.
[38] T.J.R. Hughes, The Finite Element Method. Linear Static and Dynamic Analysis, Prentice-Hall, 1987.
[39] J.L. Steger, F.C. Dougherty, J.A. Benek, A Chimera grid scheme, in: K.N. Ghia, U. Ghia (Eds.), Advances in Grid Generation, vol. ASME FED-5, 1983, pp. 59–69.
[40] J.A. Benek, P.G. Buning, J.L. Steger, A 3-D Chimera grid embedding technique, AIAA 85-1523CP, 1985.
[41] J.L. Steger, J.A. Benek, On the use of composite grid schemes in computational aerodynamics, Comput. Methods Appl. Mech. Engrg. 64 (1987) 301–320.
[42] J.A. Benek, T.L. Tonegan, N.E. Suhs, Extended Chimera grid embedding system with application to viscous flow, 1987, AIAA Paper, pp. 87–1126.
[43] E. Guilmineau, J. Piquet, P. Queutey, Two-dimensional turbulent viscous flow simulation past airfoils at fixed incidence, Comput. Fluids 26 (2) (1997) 135–162.
[44] E. Pärt-Enander, Overlapping Grids and Applications in Gas Dynamics, Ph.D. thesis, Uppsala Universitet, Sweden, 1995.
[45] J.-J. Chattot, Y. Wang, Improve treatment of interstecting bodies with the Chimera method and validation with a simple and fast flow sovler, Comput. Fluids 27 (5–6) (1998) 721–740.
[46] P.-L. Lions, On the Schwarz alternating method. II, in: T. Chan, R. Glowinski, J. Périaux, O. Widlund (Eds.), Domain Decomposition Methods, SIAM, Philadelphia, 1989, pp. 47–70.
[47] A.J. Mc Connell, Applications of Tensor Analysis, Dover, 1957.
[48] M.S. Engelman, M.A. Jamnia, Transient flow past a circular cylinder: A benchmark solution, Int. J. Numer. Meth. Fluids 11 (1990) 985–1000.
[49] R. Codina, M. Vázquez, O.C. Zienkiewicz, A general algorithm for compressible and incompressible flow-Part III. the semi-implicit form, Int. J. Numer. Meth. Fluids 27 (1998) 13–32.